The Logic of Intelligence

Pei Wang

Department of Computer and Information Sciences, Temple University
Philadelphia, PA 19122 USA
pei.wang@temple.edu - http://www.cis.temple.edu/~pwang/

Abstract

Is there an "essence of intelligence" that distinguishes intelligent systems from non-intelligent systems? If there is, then what is it? This chapter suggests an answer to these questions by introducing the ideas behind the NARS (Non-Axiomatic Reasoning System) project. NARS is based on the opinion that the essence of intelligence is the ability to adapt with insufficient knowledge and resources. According to this belief, the author has designed a novel formal logic, and implemented it in a computer system. Such a"logic of intelligence" provides a unified explanation for many cognitive functions of the human mind, and is also concrete enough to guide the actual building of a general purpose "thinking machine".

1 Intelligence and Logic

1.1 To define intelligence

The debate on the essence of intelligence has been going on for decades, and there is still little sign of consensus (this book itself is a piece of evidence).

In the "mainstream AI", the followings are some representative opinions:

> "AI is concerned with methods of achieving goals in situations in which the information available has a certain complex character. The methods that have to be used are related to the problem presented by the situation and are similar whether the problem solver is human, a Martian, or a computer program." [McCarthy, 1988]

> Intelligence usually means "the ability to solve hard problems". [Minsky, 1985]

> "By 'general intelligent action' we wish to indicate the same scope of intelligence as we see in human action: that in any real situation

behavior appropriate to the ends of the system and adaptive to the demands of the environment can occur, within some limits of speed and complexity." [Newell and Simon, 1976]

Maybe it is too early to define intelligence. It is obvious that, after the decades of study, we still do not know very much about it. There are more questions than answers. Any definition based on the current knowledge is doomed to be revised by future works. We all know that a well-founded definition is usually the result, rather than the starting point, of scientific research. However, there are still reasons for us to be concerned about the definition of intelligence at the current time. Though clarifying the meaning of a concept always helps communication, this problem is especially important for AI. As a community, AI researchers need to justify their field as a scientific discipline. Without a (relatively) clear definition of intelligence, it is hard to say why AI is different from, for instance, computer science or psychology. Is there really something novel and special, or just fancy labels on old stuff? More vitally, every researcher in the field needs to justify his/her research paradigm according to such a definition. Anyone who wants to work on artificial intelligence is facing a two-phase task: to choose a working definition of intelligence, then to produce it in a computer.

A *working definition* is a definition that is concrete enough that you can directly work with it. By accepting a working definition of intelligence, it does not mean that you really believe that it fully captures the concept "intelligence", but that you will take it as a goal for your current research project.

Therefore, the lack of a consensus on what intelligence is does not prevent each researcher from picking up (consciously or not) a working definition of intelligence. Actually, unless you keep one (or more than one) definition, you cannot claim that you are working on artificial intelligence.

By accepting a working definition of intelligence, the most important commitments a researcher makes are on the acceptable assumptions and desired results, which bind all the concrete works that follow. The defects in the definition can hardly be compensated by the research, and improper definitions will make the research more difficult than necessary, or lead the study away from the original goal.

Before studying concrete working definitions of intelligence, we need to set up a general standard for what makes a definition better than others.

Carnap met the same problem when he tried to clarify the concept "probability". The task "consists in transforming a given more or less inexact concept into an exact one or, rather, in replacing the first by the second", where the first may belong to everyday language or to a previous stage in the scientific language, and the second must be given by explicit rules for its use [Carnap, 1950].

According to Carnap, the second concept, or the *working definition* as it is called in this paper, must fulfill the following requirements [Carnap, 1950]:

1. It is *similar* to the concept to be defined, as the latter's vagueness permits.
2. It is defined in an *exact* form.
3. It is *fruitful* in the study.
4. It is *simple*, as the other requirements permit.

It seems that these requirements are also reasonable and suitable for our current purpose. Now let us see what they mean concretely to the working definition of intelligence:

Similarity (to standard usage). Though "intelligence" has no exact meaning in everyday language, it does have some common usages with which the working definition should agree. For instance, normal human beings are intelligent, but most animals and machines (including ordinary computer systems) are either not intelligent at all or much less intelligent than human beings.

Exactness (or well-definedness). Given the working definition, whether (or how much) a system is intelligent should be clearly decidable. For this reason, intelligence cannot be defined in terms of other ill-defined concepts, such as *mind, thinking, cognition, intentionality, rationality, wisdom, consciousness*, and so on, though these concepts do have close relationships with intelligence.

Fruitfulness (and instructiveness). The working definition should provide concrete guidelines for the research based on it – for instance, what assumptions can be accepted, what phenomena can be ignored, what properties are desired, and so on. Most importantly, the working definition of intelligence should contribute to the solving of fundamental problems in AI.

Simplicity. Although intelligence is surely a complex mechanism, the working definition should be simple. From a theoretical point of view, a simple definition makes it possible to explore a paradigm in detail; from a practical point of view, a simple definition is easy to use.

For our current purpose, there is no "right" or "wrong" working definition for intelligence, but there are "better" and "not-so-good" ones. When comparing proposed definitions, the four requirements may conflict with each other. For example, one definition is more fruitful, while another is simpler. In such a situation, some weighting and trade-off become necessary. However, there is no evidence showing that in general the requirements cannot be satisfied at the same time.

1.2 A working definition of intelligence

Following the preparation of the previous section, we propose here a working definition of intelligence:

> *Intelligence is the capacity of a system to adapt to its environment while operating with insufficient knowledge and resources.*

The *environment* of a system may be the physical world, or other information-processing systems (human or computer). In either case, the interactions can be described by the *experiences* (or *stimuli*) and *responses* of the system, which are streams of input and output information stream, respectively. For the system, perceivable patterns of input and producible patterns of output constitute its *interface language*.

To *adapt* means that the system learns from its experiences. It adjusts its internal structure to approach its goals, as if future situations will be similar to past situations. Not all systems adapt to their environment. For instance, a traditional computing system gets all of its knowledge during its design phase. After that, its experience does not contribute to the its behaviors. To acquire new knowledge, such a system would have to be redesigned.

Insufficient knowledge and resources means that the system works under the following restrictions:

Finite: The system has a constant information-processing capacity.
Real-time: All tasks have time requirements attached.
Open: No constraints are put on the knowledge and tasks that the system can accept, as long as they are representable in the interface language.

The two main components in the working definition, *adaptation* and *insufficient knowledge and resources*, are related to each other. An adaptive system must have some insufficiency in its knowledge and resources, for otherwise it would never need to change at all. On the other hand, without adaptation, a system may have insufficient knowledge and resources, but make no attempt to improve its capacities.

Not all systems take their own insufficiency of knowledge and resources into full consideration. Non-adaptive systems, for instance, simply ignore new knowledge in their interactions with their environment. As for artificial adaptive systems, most of them are not finite, real-time, and open, in the following senses:

1. Though all actual systems are finite, many theoretical models (for example, Turing Machine) neglect the fact that the requirements for processor time and/or memory space may go beyond the supply capacity of the system.
2. Most current AI systems do not consider time constraints at run time. Most real-time systems can handle time constraints only if they are essentially deadlines [Strosnider and Paul, 1994].
3. Various constraints are imposed on what a system can experience. For example, only questions that can be answered by retrieval and deduction from current knowledge are acceptable, new knowledge cannot conflict with previous knowledge, and so on.

Many computer systems are designed under the assumption that their knowledge and resources, though *limited* or *bounded*, are still *sufficient* to fulfill the tasks that they will be called upon to handle. When facing a situation where this assumption fails, such a system simply panics or crashes, and asks for external intervention by a human user.

For a system to work under the assumption of insufficient knowledge and resources, it should have mechanisms to handle the following types of situation, among others:

- A new processor is required when all existent processors are occupied;
- Extra memory is required when all available memory is already full;
- A task comes up when the system is busy with something else;
- A task comes up with a time requirement, so exhaustive search is not an option;
- New knowledge conflicts with previous knowledge;
- A question is presented for which no sure answer can be deduced from available knowledge;

For traditional computing systems, these types of situations usually require human intervention or else simply cause the system to refuse to accept the task or knowledge involved. However, for a system designed under the assumption of insufficient knowledge and resources, these are *normal situations*, and should be managed smoothly by the system itself. According to the above definition, intelligence is a “highly developed form of mental adaptation” [Piaget, 1960].

When defining intelligence, many authors ignore the complementary question: what is unintelligent? If everything is intelligent, then this concept is empty. Even if we agree that intelligence, like almost all properties, is a matter of degree, we still need criteria to indicate what makes a system more intelligent than another. Furthermore, for AI to be a (independent) discipline, we require the concept “intelligence” to be different from other established concepts, otherwise, we are only talking about some well-known stuff with a new name, which is not enough to establish a *new* branch of science. For example, if every computer system is intelligent, it is better to stay within the theory of computation. Intuitively, “intelligent system” does not mean a faster and bigger computer. On the other hand, an unintelligent system is not necessarily incapable or gives only wrong results. Actually, most ordinary computer systems and many animals can do something that human beings cannot. However, these abilities do not earn the title “intelligent” for them. What is missing in these capable-but-unintelligent systems? According to the working definition of intelligence introduced previously, an *unintelligent* system is one that does not adapt to its environment. Especially, in artificial systems, an *unintelligent* system is one that is designed under the assumption that it only works on problems for which the system has sufficient knowledge and resources. An intelligent system is not always “better” than an unintelligent system for practical purposes. Actually, it is the contrary: when a problem can be solved by both of them, the unintelligent system is usually better, because it guarantees a correct solution. As Hofstadter said, for tasks like adding two numbers, a “reliable but mindless” system is better than an “intelligent but fallible” system [Hofstadter, 1979].

1.3 Comparison with other definitions

Since it is impossible to compare the above definition to each of the existing working definitions of intelligence one by one, we will group them into several categories.

Generally speaking, research in artificial intelligence has two major motivations. As a field of science, we want to learn how the human mind, and “mind” in general, works; and as a branch of technology, we want to apply computers to domains where only the human mind works well currently. Intuitively, both goals can be achieved if we can build computer systems that are “similar to the human mind”. But in what sense they are “similar”? To different people, the desired similarity may involve *structure, performance, capacity, function,* or *principle.* In the following, we discuss typical opinions in each of the fivecategories, to see where these working definitions of intelligence will lead AI.

To simulate human brain Intelligence is produced by the human brain, so maybe AI should attempt to simulate a brain in a computer system as faithful as possible. Such an opinion is put in its extreme form by neuroscientists Reeke and Edelman, who argue that “the ultimate goals of AI and neuroscience are quite similar” [Reeke and Edelman, 1988].

Though it sounds reasonable to identify AI with *brain model*, few AI researchers take such an approach in a very strict sense. Even the “neural network” movement is “not focused on *neural modeling* (i.e., the modeling of neurons), but rather ... focused on *neurally inspired* modeling of cognitive processes” [Rumelhart and McClelland, 1986]. Why? One obvious reason is the daunting *complexity* of this approach. Current technology is still not powerful enough to simulate a huge neural network, not to mention the fact that there are still many mysteries about the brain. Moreover, even if we were able to build a brain model at the neuron level to any desired accuracy, it could not be called a success for AI, though it would be a success for neuroscience.

AI is more closely related to the concept “model of mind” – that is, a *high-level* description of brain activity in which biological concepts do not appear [Searle, 1980]. A high-level description is preferred, not because a low-level description is impossible, but because it is usually simpler and more general. A distinctive characteristic of AI is the attempt to “get a mind without a brain” – that is, to describe mind in a medium-independent way. This is true for all models: in building a model, we concentrate on certain properties of an object or process and ignore irrelevant aspects; in so doing, we gain insights that are hard to discern in the object or process itself. For this reason, an accurate duplication is not a model, and a model including unnecessary details is not a good model. If we agree that "brain" and "mind" are different concepts, then *a good model of brain is not a good model of mind*, though the former is useful for its own sake, and helpful for the building of the latter.

To duplicate human behaviors Given that we always judge the intelligence of other people by their behavior, it is natural to use "reproducing the behavior produced by the human mind as accurately as possible" as the aim of AI. Such a working definition of intelligence asks researchers to use the Turing Test [Turing, 1950] as a *sufficient and necessary* condition for having intelligence, and to take psychological evidence seriously.

Due to the nature of the Turing Test and the resource limitations of a concrete computer system, it is out of question for the system to have pre-stored in its memory all possible questions and proper answers in advance, and then to give a convincing imitation of a human being by searching such a list. The only realistic way to imitate human performance in a conversation is to produce the answers in real time. To do this, it not only needs cognitive faculties, but also much prior "human experience" [French, 1990]. Therefore, it must have a body that feels human, it must have all human motivations (including biological ones), and it must be treated by people as a human being – so it must simply be an "artificial human", rather than a computer system with artificial intelligence.

As French points out, by using behavior as evidence, the Turing Test is a criterion solely for *human* intelligence, not for intelligence in general [French, 1990]. Such an approach can lead to good psychological models, which are valuable for many reasons, but it suffers from "human chauvinism" [Hofstadter, 1979] – we would have to say, according to the definition, that the science-fiction alien creature E. T. was not intelligent, because it would definitely fail the Turing Test.

Though "reproducing human (verbal) behavior" may still be a sufficient condition for being intelligent (as suggested by Turing), such a goal is difficult, if not impossible, to achieve. More importantly, it is not a necessary condition for being intelligent, if we want "intelligence" to be a more general concept than "human intelligence".

To solve hard problems In everyday language, "intelligent" is usually applied to people who can solve hard problems. According to this type of definition, intelligence is the *capacity* to solve hard problems, and *how* the problems are solved is not very important.

What problems are "hard"? In the early days of AI, many researchers worked on intellectual activities like game-playing and theorem-proving. Nowadays, expert-system builders aim at "real-world problems" that crop up in various domains. The presumption behind this approach is: "Obviously, experts are intelligent, so if a computer system can solve problems that only experts can solve, the computer system must be intelligent, too". This is why many people take the success of the chess-playing computer Deep Blue as a success of AI.

This movement has drawn in many researchers, produced many practically useful systems, attracted significant funding, and thus has made important contributions to the development of the AI enterprise. Usually, the systems are developed by analyzing domain knowledge and expert strategy, then building them into a computer system. However, though often profitable, these systems do not provide much insight into how the mind works. No wonder people ask, after learning how such a system works, "Where's the AI?" [Schank, 1991] – these systems look just like ordinary computer application systems, and still suffer from great rigidity and brittleness (something AI wants to avoid).

If intelligence is defined as "the capacity to solve hard problems", then the next question is: "Hard for whom?" If we say "hard for human beings", then most existing computer software is already intelligent – no human can manage a database as well as a database management system, or substitute a word in a file as fast as an editing program. If we say "hard for computers," then AI becomes "whatever hasn't been done yet," which has been dubbed "Tesler's Theorem" [Hofstadter, 1979]. The view that AI is a "perpetually extending frontier" makes it attractive and exciting, which it deserves, but tells us little about how it differs from other research areas in computer science – is it fair to say that the problems there are easy? If AI researchers cannot identify other commonalities of the problems they attack besides mere hardness, they will be unlikely to make any progress in understanding and replicating intelligence.

To carry out cognitive functions According to this view, intelligence is characterized by a set of cognitive functions, such as reasoning, perception, memory, problem solving, language use, and so on. Researchers who subscribe to this view usually concentrate on just one of these functions, relying on the idea that research on all the functions will eventually be able to be combined, in the future, to yield a complete picture of intelligence. A "cognitive function" is often defined in a general and abstract manner. This approach has produced, and will continue to produce, tools in the form of software packages and even specialized hardware, each of which can carry out a function that is similar to certain mental skills of human beings, and therefore can be used in various domains for practical purposes. However, this kind of success does not justify claiming that it is the proper way to study AI. To define intelligence as a "toolbox of functions" has serious weaknesses.

When specified in isolation, an implemented function is often quite different from its "natural form" in the human mind. For example, to study analogy without perception leads to distorted cognitive models [Chalmers et al., 1992]. Even if we can produce the desired tools, this does not mean that we can easily combine them, because different tools may be developed under different assumptions, which prevents the tools from being combined.

The basic problem with the “toolbox” approach is: without a “big picture” in mind, the study of a cognitive function in an isolated, abstracted, and often distorted form simply does not contribute to our understanding of intelligence.

A common counterargument runs something like this: “Intelligence is very complex, so we have to start from a single function to make the study tractable.” For many systems with weak internal connections, this is often a good choice, but for a system like the mind, whose complexity comes directly from its tangled internal interactions, the situation may be just the opposite. When the so-called “functions” are actually phenomena produced by a complex-but-unified mechanism, reproducing all of them together (by duplicating the mechanism) is simpler than reproducing only one of them.

To develop new principles According to this type of opinions, what distinguishes intelligent systems and unintelligent systems are their *postulations*, applicable *environments*, and basic *principles* of information processing.

The working definition of intelligence introduced earlier belongs to this category. As a system adapting to its environment with insufficient knowledge and resources, an intelligent system should have many cognitive *functions*, but they are better thought of as emergent phenomena than as well-defined tools used by the system. By learning from its experience, the system potentially can acquire the *capacity* to solve hard problems – actually, *hard problems* are those for which a solver (human or computer) has insufficient knowledge and resources – but it has no such built-in capacity, and thus, without proper training, no capacity is guaranteed, and acquired capacities can even be lost. Because the human mind also follows the above principles, we would hope that such a system would behave similarly to human beings, but the similarity would exist at a more abstract level than that of concrete *behaviors*. Due to the fundamental difference between human experience/hardware and computer experience/hardware, the system is not expected to accurately reproduce masses of psychological data or to pass a Turing Test. Finally, although the internal *structure* of the system has some properties in common with a description of the human mind at the subsymbolic level, it is not an attempt to simulate a biological neural network.

In summary, the *structure* approach contributes to neuroscience by building brain models, the *behavior* approach contributes to psychology by providing explanations of human behavior, the *capacity* approach contributes to application domains by solving practical problems, and the *function* approach contributes to computer science by producing new software and hardware for various computing tasks. Though all of these are valuable for various reasons, and helpful in the quest after AI, these approaches do not, in my opinion, concentrate on the *essence* of intelligence.

To be sure, what has been proposed in my definition of intelligence is not entirely new to the AI community. Few would dispute the proposition that adaptation, or learning, is essential for intelligence. Moreover, “insufficient knowledge and resources" is the focus of many subfields of AI, such as heuristic search, reasoning under uncertainty, real-time planning, and machine learning. Given this situation, what is *new* in this approach? It is the following set of principles:

1. An explicit and unambiguous definition of intelligence as "adaptation under insufficient knowledge and resources".
2. A further definition of the phrase "with insufficient knowledge and resources" as *finite*, *real-time*, and *open*.
3. The design of all formal and computational aspects of the project keeping the two previous definitions foremost in mind.

1.4 Logic and reasoning system

To make our discussion more concrete and fruitful, let us apply the above working definition of intelligence to a special type of information processing system – reasoning system.

A reasoning system usually has the following components:

1. *a formal language* for knowledge representation, as well as communication between the system and its environment;
2. *a semantics* that determines the meanings of the words and the truth values of the sentences in the language;
3. *a set of inference rules* that match questions with knowledge, infer conclusions from promises, and so on;
4. *a memory* that systematically stores both questions and knowledge, and provides a working place for inferences;
5. *a control mechanism* that is responsible for choosing premises and inference rules for each step of inference.

The first three components are usually referred to as a *logic*, or the *logical part* of the reasoning system, and the last two as the *control part* of the system.

According to the previous definition, being a reasoning system is neither necessary nor sufficient for being intelligent. However, an *intelligent reasoning system* does provide a suitable framework for the study of intelligence, for the following reasons:

- It is a general-purpose system. Working in such a framework keeps us from being bothered by domain-specific properties, and also prevents us from cheating by using domain-specific tricks.
- Compared with cognitive activities like low-level perception and motor control, reasoning is at a more abstract level, and is one of the cognitive skills that collectively make human beings so qualitatively different from other animals.

- The framework of reasoning system is highly flexible and extendable. We will see that we can carry out many other cognitive activities in it when the concept of "reasoning" is properly extended.
- Most research on reasoning systems is carried out within a paradigm based on assumptions directly opposed to the one presented above. By "fighting in the backyard of the rival", we can see more clearly what kinds of effects the new ideas have.

Before showing how an intelligent reasoning system is designed, let us first see its opposite – that is, a reasoning system designed under the assumption that its knowledge and resources are *sufficient* to answer the questions asked by its environment (so no adaptation is needed). By definition, such a system has the following properties:

1. No new knowledge is necessary. All the system needs to know to answer the questions is already there at the very beginning, expressed by a set of axioms.
2. The axioms are *true*, and will remain true, in the sense that they correspond to the actual situation of the environment.
3. The system answers questions by applying a set of formal rules to the axioms. The rules are sound and complete (with respect to the valid questions), therefore they guarantee correct answers for all questions.
4. The memory of the system is so big that all axioms and intermediate results can always be contained within it.
5. There is an algorithm that can carry out any required inference in finite time, and it runs so fast that it can satisfy all time requirements that may be attached to the questions.

This is the type of system dreamed of by Leibniz, Boole, Hilbert, and many others. It is usually referred to as a "decidable axiomatic system" or a "formal system". The attempt to build such systems has dominated the study of logic for a century, and has strongly influenced the research of artificial intelligence. Many researchers believe that such a system can serve as a model of human thinking.

However, if intelligence is defined as "to adapt under insufficient knowledge and resources", what we want is the *contrary*, in some sense, to an axiomatic system, though it is still *formalized* or *symbolized* in a technical sense. That is why *Non-Axiomatic Reasoning System*, NARS for short, is chosen as the name for the intelligent reasoning system to be introduced in the following sections.

Between "pure-axiomatic" systems and "non-axiomatic" ones, there are also "semi-axiomatic" systems. They are designed under the assumption that knowledge and resources are insufficient in some, but not all, aspects. Consequently, adaptation is necessary. Most current AI approaches fall into this category. According to our working definition of intelligence, pure-axiomatic systems are not intelligent at all, non-axiomatic systems are intelligent, and semi-axiomatic systems are intelligent in certain aspects.

Pure-axiomatic systems are very useful in mathematics, where the aim is to idealize knowledge and questions to such an extent that the revision of knowledge and the deadlines of questions can be ignored. In such situations, questions can be answered in a way that is so accurate and reliable that the procedure can be reproduced by a Turing Machine. We need intelligence only when no such pure-axiomatic method can be used, due to the insufficiency of knowledge and resources. For the same reason, the performance of a non-axiomatic system is not necessarily better than that of a semi-axiomatic system, but it can work in environments where the latter cannot be used.

Under the above definitions, intelligence is still (as we hope) a matter of degree. Not all systems in the "non-axiomatic" and "semi-axiomatic" categories are equally intelligent. Some systems may be more intelligent than some other systems by having a higher resources efficiency, using knowledge in more ways, communicating with the environment in a richer language, adapting more rapidly and thoroughly, and so on.

"Non-axiomatic" does not mean "everything changes". In NARS, nothing is fixed as far as the *content* of knowledge is concerned, but as we will see in the following sections, how the changes happen is fixed, according to the inference rules and control strategy of the system, which remain constant when the system is running. This fact does not make NARS "semi-axiomatic", because the fixed part is not in the "object language" level, but in the "meta-language" level. In a sense, we can say that the "meta-level" of NARS is not non-axiomatic, but pure-axiomatic. For a reasoning system, a fixed inference rule is not the same as an axiom.

Obviously, we can allow the "meta-level" of NARS to be non-axiomatic, too, and therefore give the system more flexibility in its adaptation. However, that approach is not adopted in NARS at the current stage, for the following reasons:

- "Complete self-modifying" is an illusion. As Hofstadter put it, "below every tangled hierarchy lies an inviolate level" [Hofstadter, 1979]. If we allow NARS to modify its meta-level knowledge, i.e., its inference rules and control strategy, we need to give it (fixed) meta-meta-level knowledge to specify how the modification happens. As flexible as the human mind is, it cannot modify its own "low of thought".
- Though high-level self-modifying will give the system more flexibility, it does not necessarily make the system more intelligent. Self-modifying at the meta-level is often dangerous, and it should be used only when the same effect cannot be produced in the object-level. To assume "the more radical the changes can be, the more intelligent the system will be" is unfounded. It is easy to allow a system to modify its own source code, but hard to do it right.

- In the future, we will explore the possibility of meta-level learning in NARS, but will not attempt to do so until the object-level learning is mature. To try everything at the same time is just not a good engineering approach, and this does not make NARS less non-axiomatic, according to the above definition.

Many arguments proposed previously against logical AI [Birnbaum, 1991, McDermott, 1987], symbolic AI [Dreyfus, 1992], or AI as a whole [Searle, 1980, Penrose, 1994], are actually against a more specific target: pure-axiomatic systems. These arguments are powerful in revealing that many aspects of intelligence cannot be produced by a pure-axiomatic system (though these authors do not use this term), but some of them are misleading by using such a system as the prototype of AI research. By working on a reasoning system, with its formal language and inference rules, we do not necessarily bind ourselves with the commitments accepted by the traditional logistic AI paradigms. As we will see in the following, NARS shares more philosophical opinions with the *subsymbolic*, or *connectionist* movement [Hofstadter, 1985, Holland, 1986, Holland et al., 1986, Rumelhart and McClelland, 1986, Smolensky, 1988].

What is the relationship of artificial intelligence and computer science? What is the position of AI in the whole science enterprise? Traditionally, AI is referred to as a branch of computer science. According to our previous definitions, AI can be implemented with the tools provided by computer science, but from a *theoretical* point of view, they make opposite assumptions: computer science focuses on pure-axiomatic systems, while AI focuses on non-axiomatic systems. The fundamental assumptions of computer science can be found in mathematical logic (especially, first-order predicate logic) and computability theory (especially, Turing Machine). These theories take the sufficiency of knowledge and resources as implicit postulates, therefore adaptation, plausible inference, and tentative solutions of problems are neither necessary nor possible.

Similar assumptions are often accepted by AI researchers with the following justification: “We know that the human mind usually works with insufficient knowledge and resources, but if you want to set up a *formal* model and then a *computer* system, you must somehow *idealize* the situation.” It is true that every formal model is an idealization, and so is NARS. The problem is what to omit and what to preserve in the idealization. In the current implementation of NARS, many factors that should influence reasoning are ignored, but the insufficiency of knowledge and resources is strictly assumed throughout. Why? Because it is a *definitive* feature of intelligence, so if it were lost through the “idealization”, the resulting study would be about something else.

2 The Components of NARS

Non-Axiomatic Reasoning System (NARS) is designed to be an intelligent reasoning system, according to the working definition of intelligence introduced previously.

In the following, let us see how the major components of NARS (its formal language, semantics, inference rules, memory, and control mechanism) are determined, or strongly suggested, by the definition, and how they differ from the components of an axiomatic system. Because this chapter is concentrated in the philosophical and methodological foundation of the NARS project, formal descriptions and detailed discussions for the components are left to other papers [Wang, 1994b, Wang, 1995b, Wang, 2001a].

2.1 Experience-grounded semantics

Axiomatic reasoning systems (and most semi-axiomatic systems) use "model-theoretic semantics". Informally speaking, a model is a description of a domain, with relations among objects specified. For a reasoning system working on the domain, an "interpretation" maps the terms in the system to the objects in the model, and the predicates in the systems to the relations in the model. For a given term, its *meaning* is its image in the model under the interpretation. For a given proposition, its *truth value* depends on whether it corresponds to a fact in the model. With such a semantics, the reasoning system gets a constant "reference", the model, according to which truth and meaning within the system is determined. Though model-theoretic semantics comes in different forms, and has variations, this "big picture" remains unchanged.

This kindof semantics is not suitable for NARS. As an adaptive system with insufficient knowledge and resources, the system cannot judge the truthfulness of its knowledge against a static, consistent, complete model. Instead, truth and meaning have to be grounded on the system's *experience* [Wang, 1995a, Wang, 1995b]. Though a section of experience is also a description of the system's environment, it is fundamentally different from a model, since experience changes over time, is never complete, and often inconsistent. Furthermore, experience is directly accessible to the system, while model is often "in the eye of an observer".

According to an experience-grounded semantics, truth value becomes a function of the amount of available evidence, therefore inevitably becomes dynamic and subjective, though not arbitrary. In such a system, no knowledge is "true" in the sense that it is guaranteed to be confirmed by future experience. Instead, the truth value of a statement indicates the degree to which the statement is confirmed by past experience. The system will use such knowledge to predict the future, because it is exactly what "adaptive", and therefore "intelligent", means. In this way, "truth" means quite different (though closely related) things in non-axiomatic systems and axiomatic systems.

Similarly, the *meaning* of a term, that is, what makes the term different from other terms to the system, is determined by its relationships to other terms, according to the experience, rather than by an interpretation that maps it into an object in a model.

With insufficient resources, the truth-value of each statement and the meaning of each term in NARS is usually grounded on part of the experience. As a result, even without new experience, the inference activity of the system will change the truth-values and meanings, by taking previously available-but-ignored experience into consideration. In contrary, according to model-theoretic semantics, the internal activities of a system have no effects on truth value and meaning of the language it uses.

"Without an interpretation, a system has no access to the semantics of a formal language it uses" is the central argument in Searle's "Chinese room" thought experiment against strong AI [Searle, 1980]. His argument is valid for model-theoretic semantics, but not for experience-grounded semantics. For an intelligent reasoning system, the latter is more appropriate.

2.2 Inheritance statement

As discussed above, "adaptation with insufficient knowledge and resources" demands an experience-grounded semantics, which in turn requires a formal knowledge representation language in which *evidence* can be naturally defined and measured.

For a non-axiomatic reasoning system, it is obvious that binary truth value is not enough. With past experience as the only guidance, the system not only needs to know whether there is counter example (negative evidence), but also needs to know its amount, with respect to the amount of positive evidence. To have a general, domain-independent method to compare competing answers, a numerical truth value, or a *measurement of uncertainty*, becomes necessary for NARS, which quantitatively records the relationship between a statement and available evidence. Furthermore, "positive evidence" and "irrelevant stuff" need to be distinguished from each other, too.

Intuitively speaking, the simplest case to define evidence is for a general statement about many cases, while some of them are confirmed by past experience (positive evidence), and some others are disconfirmed by past experience (negative evidence). Unfortunately, the most popular formal language for knowledge representation, first-order predicate calculus, cannot be easily used in this way. In this language, a "general statement", such as "Ravens are black", is represented as a "universal proposition", such as $(\forall x)(Raven(x) \rightarrow Black(x))$. In the original form of first-order predicate calculus, there is no such a notion as "evidence", and the proposition is either true or false, depending on whether there is such an object x in the domain that makes $Raven(x)$ true and $Black(x)$ false. It is natural to define constants that make the proposition true as its positive evidence, and the constants that make it false its negative evidence. However, such a naive solution has serious problems [Wang, 1999, Wang, 2001c]:

- Only the existence of negative evidence contribute to the truth value of the universal proposition, while whether there is "positive evidence" does not matter. This is where Popper's refutation theory [Popper, 1959] came from.
- Every constant is either a piece of positive evidence or a piece of negative evidence, and nothing is irrelevant. This is where Hempel's conformation paradox [Hempel, 1943] came from.

Though evidence is hard to define in (first-order) predicate calculus, it is easy to do in a properly designed *categorical logic*. Categorical logic, or "term-oriented" logic, is another family of formal logic, exemplified by Aristotle's Syllogism [Aristotle, 1989]. The major formal features that distinguish it from predicate logic is the use of subject-predicate statements and syllogistic inference rules. Let us start with the first feature.

NARS uses a categorical language that is based on an *inheritance* relation, "$\rightarrow$". The relation, in its ideal form, is a reflexive and transitive binary relation defined on *terms*, where a term can be thought as the name of a concept. For example, "$raven \rightarrow bird$" is an inheritance statement with "*raven*" as *subject term* and "*bird*" as *predicate term*. Intuitively, it says that the subject is a *specialization* of the predicate, and the predicate is a *generalization* of the subject. The statement roughly corresponds to the English sentence "Raven is a kind of bird". Based on the inheritance relation, the *extension* and *intension* of a term are defined as the set of its specializations and generalizations, respectively. That is, for a given term T, its extension T^E is the set $\{x \mid x \rightarrow T\}$, and its intension T^I is the set $\{x \mid T \rightarrow x\}$. Given the reflexivity and transitivity of the inheritance relation, it can be proven that for any terms S and P, $S \rightarrow P$ is true if and only if S^E is included in P^E, and P^I is included in S^I. In other words, "There is an inheritance relation from S to P" is equivalent to "P inherits the extension of S, and S inherits the intension of P".

When considering "imperfect" inheritance statements, the above theorem naturally gives us the definition of (positive and negative) evidence. For a given statement "$S \rightarrow P$", if a term M in both S^E and P^E, or in both P^I and S^I, then it is a piece of positive evidence for the statement, because as far as M is concerned, the proposed inheritance is true; if M in S^E but not in P^E, or in P^I but not in S^I, then it is a piece of negative evidence for the statement, because as far as M is concerned, the proposed inheritance is false; if M is neither in S^E nor in P^I, it is not evidence for the statement, and whether it is also in P^E or S^I does not matter. Let us use w^+, w^-, and w for the amount of positive, negative, and total evidence, respectively, then we have $w^+ = |S^E \cap P^E| + |P^I \cap S^I|$, $w^- = |S^E - P^E| + |P^I - S^I|$, $w =$

$w^+ + w^- = |S^E| + |P^I|$. Finally, we define the truth value of a statement to be a pair of numbers $<f, c>$. Here f is called the *frequency* of the statement, and $f = w^+/w$. The second component c is called the *confidence* of the statement, and $c = w/(w+k)$, where k is a system parameter with 1 as the default value. For a more detailed discussion, see [Wang, 2001b].

Now we have got the technical basic of the experience-grounded semantics: If the experience of the system is a set of inheritance statements defined above, then for any term T, we can determine its *meaning*, which is its extension and intension (according to the experience), and for any inheritance statement $S \rightarrow P$, we can determine its positive evidence and negative evidence (by comparing the meaning of the two terms), then calculate its truth value according to the above definition.

Of course, the actual experience of NARS is not a set of binary inheritance statements, nor does the system determine the truth value of a statement in the above way. The actual experience of NARS is a stream of statements, with their truth values, represented by the $<f, c>$ pair. Within the system, new statements are derived by the inference rules, with truth-value functions calculating the truth values of the conclusions from those of the premises. The purpose of the above definitions is to *define* the truth value in an idealized situation, and to provide a foundation for the truth value functions (to be discussed in the following).

2.3 Categorical language

Based on the inheritance relation introduced above, NARS uses a powerful "categorical language", obtained by extending the above core language in various directions:

Derived inheritance relations: Beside the *inheritance* relation defined previously, NARS also includes several of its variations. For example,

- the *similarity* relation $\leftrightarrow$ is symmetric inheritance;
- the *instance* relation $\circ\!\!\rightarrow$ is an inheritance relation where the subject term is treated as an atomic instance of the predicate term;
- the *property* relation $\rightarrow\!\!\circ$ is an inheritance relation where the predicate term is treated as a primitive property of the subject term.

Compound terms: In inheritance statements, the (subject and predicate) terms not only can be simple names (as in the above examples), but also can be compound terms formed by other terms with logical operator. For example, if A and B are terms, we have

- their *union* $(A \cup B)$ is a compound term, initially defined by $(A \cup B)^E = (A^E \cup B^E)$ and $(A \cup B)^I = (A^I \cap B^I)$;
- their *conjunction* $(A \cap B)$ is a compound term, initially defined by $(A \cap B)^E = (A^E \cap B^E)$ and $(A \cap B)^I = (A^I \cup B^I)$.

With compound terms, the expressive power of the language is greatly extended.

Ordinary relation: In NARS, only the inheritance relation and its variations are defined as logic constants that are directly recognized by the inference rules. All other relations are converted into inheritance relations with compound terms. For example, an arbitrary relation R among three terms A, B, and C is usually written as $R(A, B, C)$, which can be equivalently rewritten as one of the following inheritance statements (i.e., they have the same meaning and truth value):

- $(A, B, C) \circ\!\!\rightarrow R$, where the subject term is a compound (A, B, C), an ordered tuple. This statement says "The relation among A, B, C (in that order) is an instance of the relation R."
- $A \circ\!\!\rightarrow R(*, B, C)$, where the predicate term is a compound $R(*, B, C)$ with a "wildcard", $*$. This statement says "A is such an x that satisfies $R(x, B, C)$."
- $B \circ\!\!\rightarrow R(A, *, C)$. Similarly, "$B$ is such an x that satisfies $R(A, x, C)$."
- $C \circ\!\!\rightarrow R(A, B, *)$. Again, "$C$ is such an x that satisfies $R(A, B, x)$."

Higher-order term: In NARS, a statement can be used as a term, which is called a "higher-order" term. For example, "Bird is a kind of animal" is represented by statement $bird \rightarrow animal$, and "Tom knows that bird is a kind of animal" is represented by statement $(bird \rightarrow animal) \circ\!\!\rightarrow know(Tom, *)$, where the subject term is a statement. Compound higher-order terms are also defined: if A and B are higher-order terms, so do their negations ($\neg A$ and $\neg B$), disjunction ($A \vee B$), and conjunction ($A \wedge B$).

Higher-order relation: Higher-order relations are the ones whose subject term and predicate term are both higher-order terms. In NARS, there are two of them defined as logic constants:

- *implication*, $\Rightarrow$, which is intuitively correspond to "if-then", and is defined as isomorphic to *inheritance*, $\rightarrow$;
- *equivalence*, $\Leftrightarrow$, which is intuitively correspond to "if-and-only-if", and is defined as isomorphic to *similarity*, $\leftrightarrow$.

Non-declarative sentences: Beside the various types statements introduced above, which represent the system's declarative knowledge, the formal language of NARS uses similar formats to represent non-declarative sentences:

- a *question* is either a statement whose truth value needs to be evaluated ("yes/no" questions), or a statement containing variables to be instantiated ("what" questions);
- a *goal* is a statement whose truthfulness needs to be established by the system through the execution of relevant operations.

For each type of statements, its truth value is defined similarly to how we define truth value for inheritance statement.

With the above structures, the expressive power of the language is roughly the same as a typical natural language (such as English or Chinese). There is no one-to-one mapping between sentences in this language and those in first-order predicate calculus, though approximate mapping is possible for many sentences. While first-order predicate calculus may still be better to represent mathematical knowledge, this new language will be better to represent empirical knowledge.

2.4 Syllogistic inference rules

Due to insufficient knowledge, the system needs to do non-deductive inference, such as induction, abduction, and analogy, to extend past experience to novel situations. In this context, deductions become fallible, too, in the sense that the conclusions may be revised by new knowledge, even if the premises remain unchallenged. According to the experience-grounded semantics, the definition of *validity* of inference rules is changed. Instead of generating infallible conclusions, a valid rule should generate conclusions whose truth value is evaluated against (and only against) the evidence provided by the premises.

As mentioned previously, a main feature that distinguish categorical logics from predicate/propositional logics is the use of *syllogistic* inference rules, each of which takes a pair of premises that share a common term. For inference among inheritance statements, there are three possible combinations if the two premises share exactly one term:

$$\begin{array}{ccc}
\textbf{deduction} & \textbf{induction} & \textbf{abduction} \\
M \rightarrow P <f_1, c_1> & M \rightarrow P <f_1, c_1> & P \rightarrow M <f_1, c_1> \\
S \rightarrow M <f_2, c_2> & M \rightarrow S <f_2, c_2> & S \rightarrow M <f_2, c_2> \\
\hline
S \rightarrow P <f, c> & S \rightarrow P <f, c> & S \rightarrow P <f, c>
\end{array}$$

Each inference rule has its own truth-value function to calculate the truth value of the conclusion according to those of the premises. In NARS, these functions are designed in the following way:

1. Treat all relevant variables as binary variables taking 0 or 1 values, and determine what values the conclusion should have for each combination of premises, according to the semantics.
2. Represent the truth values of conclusion obtained above as Boolean functions of those of the premises.
3. Extend the Boolean operators into real number functions defined on [0, 1] in the following way:

$$\begin{aligned}
NOT(x) &= 1 - x \\
AND(x_1, ..., x_n) &= x_1 * ... * x_n \\
OR(x_1, ..., x_n) &= 1 - (1 - x_1) * ... * (1 - x_n)
\end{aligned}$$

4. Use the extended operators, plus the relationship between truth value and amount of evidence, to rewrite the above functions.

The result is the following:

deduction	**induction**	**abduction**
$f = f_1 f_2/(f_1 + f_2 - f_1 f_2)$	$f = f_1$	$f = f_2$
$c = c_1 c_2 (f_1 + f_2 - f_1 f_2)$	$c = f_2 c_1 c_2/(f_2 c_1 c_2 + 1)$	$c = f_1 c_1 c_2/(f_1 c_1 c_2 + 1)$

When the two premises have the same statement, but comes from different sections of the experience, the revision rule is applied to merge the two into a summarized conclusion:

revision

$$\frac{\begin{array}{l} S \rightarrow P <f_1, c_1> \\ S \rightarrow P <f_2, c_2> \end{array}}{S \rightarrow P <f, c>}$$

Since in revision the evidence for the conclusion is the sum of the evidence in the premises, the truth value function is

$$f = \frac{f_1 c_1/(1-c_1) + f_2 c_2/(1-c_2)}{c_1/(1-c_1) + c_2/(1-c_2)}$$

$$c = \frac{c_1/(1-c_1) + c_2/(1-c_2)}{c_1/(1-c_1) + c_2/(1-c_2) + 1}$$

Beside the above four basic inference rules, in NARS there are inference rules for the variations of inheritance, as well as for the formation and transformation of the various compound terms. The truth-value functions for those rules are similarly determined.

Beside the above *forward* inference rules by which new knowledge is derived existing knowledge, NARS also has *backward* inference rules, by which a piece of knowledge is applied to a question or a goal. If the knowledge happens to provide an answer for the question or an operation to realize the goal, it is accepted as a tentative solution, otherwise a derived question or goal may be generated, whose solution, combined with the knowledge, will provide a solution to the original question or goal. Defined in this way, for each forward rule, there is a matching backward rule. Or, conceptually, we can see them as two ways to use the same rule.

2.5 Controlled concurrency in dynamic memory

As an open system working in real-time, NARS accepts new tasks all the time. A new task may be a piece of knowledge to be digested, a question to be answered, or a goal to be achieved. A new task may come from a human user or from another computer system.

Since in NARS no knowledge is absolutely true, the system will try to use as much knowledge as possible to process a task, so as to provide a better (more confident) solution. On the other hand, due to insufficient resources, the system cannot use all relevant knowledge for each task. Since new tasks come from time to time, and the system generates derived tasks constantly, at any moment the system typically has a large amount of tasks to process. For this situation, to set up a static standard for a satisfying solution [Strosnider and Paul, 1994] is too rigid a solution, because no matter how careful the standard is determined, sometimes it will be too high, and sometimes too low, given the ever changing resources demand of the existing tasks. What NARS does is to try to find the best solution given the current knowledge and resources restriction [Wang, 1995b] – similar to what an "anytime algorithm" does [Dean and Boddy, 1988].

"Bag" is a data structure specially designed in NARS for resource allocation. A bag can contain certain type of items with a constant capacity, and maintains a priority distribution among the items. There are three major operations defined on bag:

- Put an item into the bag, and if the bag is already full, remove an item with the lowest priority.
- Take an item out of the bag by key.
- Take an item out of the bag by priority, that is, the probability for an item to be selected is proportional to its priority value.

Each of the operations takes about constant time to finish, independent to the number of items in the bag.

NARS organizes knowledge and tasks into *concepts*. In the system, each term T has a corresponding concept C_T, which contains all the knowledge and tasks in which T is the subject term or predicate term. For example, knowledge $bird \rightarrow animal < 1, 0.9 >$ is stored within concept *bird* and concept *animal*. In this way, the memory of NARS can be seen roughly as a *bag* of concepts, and each concept is named by a (simple or compound) term, and contains a *bag* of knowledge and a *bag* of tasks, all of them are directly about the term.

NARS runs by repeatedly carrying out the following working cycle:

1. Take a concept from the memory by priority.
2. Take a task from the task bag of the concept by priority.
3. Take a piece of knowledge from the knowledge bag of the concept by priority.
4. According to the combination of the task and the knowledge, call the applicable inference rules on them to derive new tasks and new knowledge – in categorical logic, every inference step happens within a concept.
5. Adjust the priority of the involved task, knowledge, and concept, according to how they behave in this inference step, then put them back into the corresponding bags.

6. Put the new (input or derived) tasks and knowledge into the corresponding bags. If certain new knowledge provides the best solution so far for a user-assigned task, report a solution.

The priority value of each item reflects the amount of resources the system plans to spend on it in the near future. It has two factors:

long-term factor: The system gives higher priority to more *important* items, evaluated according to past experience. Initially, the user can assign priority values to the input tasks to indicate their relative importance, which will in turn determine the priority value of the concepts and knowledge generated from it. After each inference step, the involved items have their priority values adjusted. For example, if a piece of knowledge provides a best-so-far solution for a task, then the priority value of the knowledge is increased (so that it will be used more often in the future), and the priority value of the task is decreased (so that less time will be used on it in the future).

short-term factor: The system gives higher priority to more *relevant* items, evaluated according to current context. When a new task is added into the system, the directly related concepts are *activated*, i.e., their priority values are increased. On the other hand, the priority values decay over time, so that if a concept has not be relevant for a while, it becomes less active.

In this way, NARS process many tasks in parallel, but with different speed. This "controlled concurrency" control mechanism is similar to Hofstadter's "parallel terraced scan" strategy [Hofstadter, 1984]. Also, how a task is processed depends on the available knowledge and the priority distribution among concepts, tasks, and knowledge. Since these factors change constantly, the solution a task gets is context-dependent.

3 The Properties of NARS

As a project aimed at general-purpose artificial intelligence, NARS addresses many issues in AI and cognitive science. Though it is similar to many other approaches here or there, the project as a whole is unique in its theoretical foundation and major technical components. Designed as above, NARS shows many properties that make it more similar to human reasoning than to other AI systems.

3.1 Reasonable solutions

With insufficient knowledge and resources, NARS cannot guarantee that all the solutions it generates for tasks are *correct* in the sense that they will not be challenged by the system's future experience. Nor can it guarantee that the solutions are *optimum* given all the knowledge the system has at the moment. However, the solutions are *reasonable* in the sense that they are the best summaries of the past experience, given the current resources supply. This is similar to Good's "Type II rationality" [Good, 1983].

NARS often makes "reasonable mistakes" that are caused by the insufficiency of knowledge and resources. They are reasonable and inevitable given the working condition of the system, and they are not caused by the errors in the designing or functioning of the system.

A conventional algorithm provides a single solution to each problem, then stops working on the problem. In contrary, NARS may provide no, one, or more than one solution to a task – it reports every solution that is the best it finds, then looks for a better one (if resources are still available). Of course, eventually the system will end its processing of the task, but the reason is neither that a satisfying solution has been found, nor that a deadline is reached, but that the task has lost in the resources competition.

Like trial-and-error procedures [Kugel, 1986], NARS can "change its mind". Because truth values are determined according to experience, a later solution is judged as "better" than a previous one, because it is based on more evidence, though it is not necessarily "closer to the objective fact".

When a solution is found, usually there is no way to decide whether it is the last the system can get. In NARS, there is no "final solution" that cannot be updated by new knowledge and/or further consideration, because all solutions are based on partial experience of the system. This *self-revisable* feature makes NARS a more general model than the various *non-monotonic logics*, in which only binary statements are processed, and only the conclusions derived from default rules can be updated, but the default rules themselves are not effected by the experience of the system [Reiter, 1987].

3.2 Unified uncertainty processing

As described previously, in NARS there are various types of uncertainty, in concepts, statements, inference rules, and inference processes. NARS has a unified uncertainty measurement and calculation sub-system.

What makes this approach different from other proposed theories on uncertainty is the experience-grounded semantics. According to it, all uncertainty comes from the insufficiency of knowledge and resources. As a result, the evaluation of uncertainty is changeable and context-dependent.

From our previous definition of truth value, it is easy to recognize its relationship with probability theory. Under a certain interpretation, the frequency measurement is similar to probability, and the confidence measurement is related to the size of sample space. If this is the case, why not directly use probability theory to handle uncertainty?

Let us see a concrete case. The deduction rule takes $M \rightarrow P < f_1, c_1 >$ and $S \rightarrow M < f_2, c_2 >$ as premises, and derives $S \rightarrow P < f, c >$ as conclusion. A direct way to apply probability theory would be treating each term as a set, then turning the rule into one that calculates conditional probability $Pr(P|S)$ from $Pr(P|M)$ and $Pr(M|S)$ plus additional assumptions about the probabilistic distribution function $Pr()$. Similarly, the sample size of the conclusion would be estimated, which gives the confidence value.

Such an approach cannot be applied in NARS for several reasons:

- For an inheritance relation, evidence is defined both extensionally and intensionally, so the frequency of $S \rightarrow P$ cannot be treated as $Pr(P|S)$, since the latter is pure extensional.
- Each statement has its own evidence space, defined by the extension of its subject and the intension of its predicate.
- Since pieces of knowledge in input may come from different sources, they may contain inconsistency.
- When new knowledge comes, usually the system cannot afford the time to update all of the previous beliefs accordingly.

Therefore, though each statement can be treated as a probabilistic judgment, different statements correspond to different evidence space, and their truth values are evaluated against different bodies of evidence. As a result, they correspond to different probability distributions. For example, if we treat frequency as probability, the deduction rule should calculate $Pr_3(S \rightarrow P)$ from $Pr_1(M \rightarrow P)$ and $Pr_2(S \rightarrow M)$. In standard probability theory, there is few result that can be applied to this kind of cross-distribution calculation.

NARS solves this problem by going beyond probability theory, though still sharing certain intuition and result with it [Wang, 2001b].

In NARS, the amount of evidence is defined in such a way that it can be used to indicate *randomness* (see [Wang, 1993] for a comparison with Bayesian network [Pearl, 1988]), *fuzziness* (see [Wang, 1996] for a comparison with fuzzy logic [Zadeh, 1965]), and *ignorance* (see [Wang, 1994a] for a comparison with Dempster-Shafer theory [Shafer, 1976]). Though different types of uncertainty have different origins, they usually co-exist, and are tangled with one another in practical situations. Since NARS makes no restrictions on what can happen in its experience, and needs to make justifiable decisions when the available knowledge is insufficient, such a unified measurement of uncertainty is necessary.

There may be belief conflicts in NARS, in the sense that the same statement is assigned different truth values when derived from different parts of the experience. With insufficient resources, NARS cannot find and eliminate all implicit conflicts within its knowledge base. What it can do is, when a conflict is found, to generate a summarized conclusion whose truth value reflects the combined evidence. These conflicts are normal, rather than exceptional. Actually, their existence is a major driving force of learning, and only by their solutions some types of inference, like induction and abduction, can have their results accumulated [Wang, 1994b]. In first-order predicate logic, a pair of conflicting propositions implies all propositions. This does not happen in a categorical logic like NARS, because in categorical logics the conclusions and premises must have shared terms, and statements with the same truth value cannot substitute one another in a derivation (as does in predicate logic). As a result, NARS tolerates implicitly conflicting beliefs, and resolves explicit conflicts by evidence combination.

The concepts in NARS is uncertain because the meaning of a concept is not determined by an interpretation that links it to an external object, but by its relations with other concepts. The relations are in turn determined by the system's experience and its processing on the experience. When a concept is involved in the processing of a task, usually only part of the knowledge associated with the concept is used. Consequently, concepts become "fluid" [Hofstadter and Mitchell, 1994]:

1. No concept has a clear-cut boundary. Whether a concept is an instance of another concept is a matter of degree. Therefore, all the concepts in NARS are "fuzzy".
2. The membership evaluations are revisable. The priority distribution among the relations from a concept to other concepts also changes from time to time. Therefore, what a concept actually means to the system is variable.
3. However, the meaning of a concept is not arbitrary or random, but relatively stable, bounded by the system's experience.

3.3 NARS as a parallel and distributed network

Though all the previous descriptions present NARS as a reasoning system with formal language and rules, in fact the system can also be described as a network. We can see each term as a *node*, and each statement as a *link* between two nodes, and the corresponding truth value as the *strength* of the link. Priorities are defined among nodes and links. In each inference step, two adjacent links generate new links, and different types of inference correspond to different combinations of the links [Minsky, 1985, Wang, 1994b]. To answer a question means to determine the strength of a link, given its beginning and ending node, or to locate a node with the strongest link from or to a given node. Because by applying rules, the topological structure of the network, the strength of the links, and the priority distribution are all changed, what the system does is much more than searching a static network for the desired link or node.

Under such an interpretation, NARS shows some similarity to the other network-based AI approaches, such as the connectionist models.

Many processes coexist at the same time in NARS. The system not only processes input tasks in parallel, but also does so for the derived subtasks. The fact that the system can be implemented in a single-processor machine does not change the situation, because what matters here is not that the processes run exactly at the same time on several pieces of hardware (though it is possible for NARS to be implemented in a multiple-processor system), but that they are not run in an one-by-one way, that is, one process begins after another ends.

Such a parallel processing model is adopted by NARS, because given the insufficiency of knowledge and resources, as well as the dynamic nature of the memory structure and resources competition, it is impossible for the system to process tasks one after another.

Knowledge in NARS is represented *distributedly* in the sense that there is no one-to-one correspondence between the input/output in the experience/response and the knowledge in the memory [Hinton et al., 1986]. When a piece of new knowledge is provided to the system, it is not simply inserted into the memory. Spontaneous inferences will happen, which generate derived conclusions. Moreover, the new knowledge may be revised when it is in conflict with previous knowledge. As a result, the coming of new knowledge may cause non-local effects in memory.

On the other hand, the answer of a question can be generated by non-local knowledge. For example, in answering the question "Is dove a kind of bird?", a piece of knowledge "$dove \rightarrow bird$" (with its truth value) stored in concepts *dove* and *bird* provides a ready-made answer, but the work does not stop. Subtasks are generated (with lower priority) and sent to related concepts. Because there may be implicit conflicts within the knowledge base, the previous "local" answer may be revised by knowledge stored somewhere else.

Therefore, the digestion of new knowledge and the generation of answers are both non-local events in memory, though the concepts corresponding to terms that appear directly in the input knowledge/question usually have larger contributions. How "global" such an event can be is determined both by the available knowledge and the resources allocated to the task.

In NARS, information is not only stored distributedly and with duplications, but also processed through multiple pathways. With insufficient knowledge and resources, when a question is asked or a piece of knowledge is told, it is usually impossible to decide whether it will cause redundancy or what is the best method to process it, so multiple copies and pathways become inevitable. Redundancy can help the system recover from partial damage, and also make the system's behaviors depend on statistical events. For example, if the same question is repeatedly asked, it will get more processor time.

Unlike many symbolic AI systems, NARS is not "brittle" [Holland, 1986] – that is, being easily "killed" by improper inputs. NARS is open and domain-independent, so any knowledge and question, as long as they can be expressed in the system's interface language, can be accepted by the system. The conflict between new knowledge and previous knowledge will not cause the "implication paradox" (i.e., from an inconsistence, any propositions can be derived). All mistakes in input knowledge can be revised by future experience to various extents. The questions beyond the system's current capacity will no longer cause a "combinatorial explosion", but will be abandoned gradually by the system, after some futile efforts. In this way, the system may fail to answer a certain question, but such a failure will not cause a paralysis.

According to the working manner of NARS, each concept as a processing unit only takes care of its own business, that is, only does inferences where the concept is directly involved. As a result, the answering of a question is usually the cooperation of several concepts. Like in connectionist models [Rumelhart and McClelland, 1986], there is no "global plan" or "central process" that is responsible for each question. The cooperation is carried out by message-passing among concepts. The generating of a specific solution is the *emergent result* of lots of local events, not only caused by the events in its derivation path, but also by the activity of other tasks that adjust the memory structure and compete for the resources. For this reason, each event in NARS is influenced by all the events that happen before it.

What directly follows from the above properties is that the solution to a specific task is context-sensitive. It not only depends on the task itself and the knowledge the system has, but also depends on how the knowledge is organized and how the resources are allocated at the moment. The *context* under which the system is given a task, that is, what happens before and after the task in the system's experience, strongly influences what solution the task receives. Therefore, if the system is given the same task twice, the solutions may be (though not necessarily) different, even though there is no new knowledge provided to the system in the interval. Here "context" means the current working environment in which a task is processed. Such contexts are dynamic and continuous, and they are not predetermined situations indexed by labels like "bank" and "hotel".

3.4 Resources competition

The system does not treat all processes as equal. It distributes its resources among the processes, and only allows each of them to progress at certain speed and to certain "depth" in the knowledge base, according to how much resources it has. Also due to insufficient knowledge, the resource distribution is maintained dynamically (adjusted while the processes are running), rather than statically (scheduled before the processes begin to run), because the distribution depends on how they work.

As a result, the processes compete with one another for resources. To speed up one process means to slow down the others. The priority value of a task reflects its (relative) priority in the competition, but does not determine its (absolute) actual resources consumption, which also depends on the priority values of the other coexisting tasks.

With insufficient processing time, it is inefficient for all the knowledge and questions to be equally treated. In NARS, some of them (with higher priority values) get more *attention*, that is, are more active or accessible, while some others are temporarily forgotten. With insufficient memory space, some knowledge and questions will be permanently forgotten – eliminated from the memory. Like in human memory [Medin and Ross, 1992], in NARS forgetting is not a deliberate action, but a side-effect caused by resource competition.

In traditional computing systems, how much time is spent on a task is determined by the system designer, and the user provides tasks at run time without time requirements. On the other hand, many real-time systems allow users to attach a deadline to a task, and the time spent on the task is determined by the deadline [Strosnider and Paul, 1994]. A variation of this approach is that the task is provided with no deadline, but the user can interrupt the process at any time to get a best-so-far answer [Boddy and Dean, 1994].

NARS uses a more flexible method to decide how much time is spent on a task, and both the system and the user influence the decision. The user can attaches an initial priority value to a task, but the actual allocation also depends on the current situation of the system, as well as on how well the task processing goes. As a result, the same task, with the same initial priority, will get more processing when the system is "idle" than when the system is "busy".

3.5 Flexible behaviors

How an answer is generated is heavily dependent on what knowledge is available and how it is organized. Facing a task, the system does not choose a method first, then collect knowledge accordingly, but lets it interact with available knowledge. In each inference step, what method is used to process a task is determined by the type of knowledge that happens to be picked up at that point.

As a result, the processing path for a task is determined dynamically at run time, by the current memory structure and resource distribution of the system, not by a predetermined problem-oriented algorithm. In principle, the behavior of NARS is unpredictable from an input task along, though still predictable from the system's initial state and complete experience.

For practical purposes, the behavior of NARS is not accurately predictable to a human observer. To exactly predict the system's solution to a specific task, the observer must know all the details of the system's initial state, and closely follow the system's experience until the solution is actually produced. When the system is complex enough (compared with the information processing capacity of the predictor), nobody can actually do these. However, it does not mean that the system works in a random manner. Its behaviors are still determined by its initial state and experience, so approximate predictions are possible.

If NARS is implemented in a von Neumann computer, can it go beyond the scope of computer science? Yes, it is possible because a computer system is a hierarchy with many levels [Hofstadter, 1979]. Some critics implicitly assume that because a certain level of a computer system can be captured by first-order predicate logic and Turing machine, these theories also bind all the performances the system can have [Dreyfus, 1992, Penrose, 1994]. This is not the case. When a system A is implemented by a system B, the former does not necessarily inherit all properties of the latter. For example, we cannot say that a computer cannot process decimal numbers (because they are implemented by binary numbers), cannot process symbols (because they are coded by digits), or cannot use functional or logical programming language (because they are eventually translated into procedural machine language).

Obviously, with its fluid concepts, revisable knowledge, and fallible inference rules, NARS breaks the regulations of classic logics. However, as a virtual machine, NARS can be based on another virtual machine which is a pure-axiomatic system, as shown by its implementation practice, and this fact does not make the system "axiomatic". If we take the system's complete *experience* and *response* as input and output, then NARS is still a Turing Machine that definitely maps inputs to outputs in finite steps. What happens here has been pointed out by Hofstadter as "something can be computational at one level, but not at another level" [Hofstadter, 1985], and by Kugel as "cognitive processes that, although they involve more than computing, can still be modelled on the machines we call 'computers' " [Kugel, 1986]. On the contrary, traditional computer systems are Turing Machines either globally (from experience to response) or locally (from question to answer).

3.6 Autonomy and creativity

The global behavior NARS is determined by the "resultant of forces" of its internal tasks. Initially, the system is driven only by input tasks. The system then derives subtasks recursively by applying inference rules to the tasks and available knowledge.

However, it is not guaranteed that the achievement of the derived tasks will turn out to be really helpful or even related to the original tasks, because the knowledge, on which the derivation is based, is revisable. On the other hand, it is impossible for the system to always determine correctly which tasks are more closely related to the original tasks. As a result, the system's behavior will to a certain extent depend on "its own tasks", which are actually more or less independent of the original processes, even though historically derived from them. This is the *functional autonomy* phenomena [Minsky, 1985]. In the extreme form, the derived tasks may become so strong that they even prevent the input tasks from being fulfilled. In this way, the derived tasks are *alienated*.

The alienation and unpredictability sometimes result in the system to be "out of control", but at the same time, they lead to *creative and original* behaviors, because the system is pursuing goals that are not directly assigned by its environment or its innateness, with methods that are not directly deduced from given knowledge.

By *creativity*, it does not mean that all the results of such behaviors are of benefit to the system, or excellent according to some outside standards. Nor does it mean that these behaviors come from nowhere, or from a "free will" of some sort. In contrary, it means that the behaviors are novel to the system, and cannot be attributed either to the designer (who determines the system's initial state and skills) or to a tutor (who determines part of the system's experience) alone. Designers and tutors only make the creative behaviors possible. What turns the possibility into reality is the system's experience, and for a system that lives in a complex environment, its experience is not completely determined by any other systems (human or computer). For this reason, these behaviors, with their results, are better to be attributed to the system *itself*, than to anyone else [Hofstadter, 1979].

Traditional computer systems always repeat the following "life cycle":

- waiting for tasks
- accepting a task
- working on the task according to an algorithm
- reporting a solution for the task
- waiting for tasks
- $\cdots$

In contrary, NARS has a "life-time of its own" [Elgot-Drapkin et al., 1991]. When the system is experienced enough, there will be lots of tasks for the system to process. On the other hand, new input can come at any time. Consequently, the system's history is no longer like the previous loop. The system usually works on its "own" tasks, but at the same time, it is always ready to respond to new tasks provided by the environment. Each piece of input usually attracts the system's attention for a while, and also causes some long-term effects. The system never reaches a "final state" and stops there, though it can be reset by a human user to its initial state. In this way, each task-processing activity is part of the system's life-time experience, and is influenced by the other activities. In comparison with NARS, traditional computer systems take each problem-solving activity as a separate life cycle with a predetermined end.

4 Conclusions

The key difference between NARS and the mainstream AI projects is not in the technical details, but in the philosophical and methodological position. The NARS project does not aim at a certain practical problem or cognitive function, but attempts to build a general-purpose intelligent system by identifying the "essence of intelligence", i.e., the underlying information processing principle, then designing the components of the system accordingly.

As described above, in the NARS project, it is assumed that "intelligence" means "adaptation with insufficient knowledge and resources", then reasoning system is chosen as the framework to apply this assumption. When designing the system, we found that all relevant traditional theories (including first-order predicate logic, model theory, probability theory, computability theory, computational complexity theory, ...) are inconsistent with the above assumption, so all major components need to be redesigned. These components, though technically simple, are fundamentally different from the traditional components in nature.

Built in this way, NARS provides a unified model for many phenomena observed in human cognition. It achieves this not by explicitly fitting psychological data, but by reproducing them from a simple and unified foundation. In this way, we see that these phenomena share a common functional explanation, and all intelligent systems, either natural or artificial, will show these phenomena as long as it is an adaptive system working with insufficient knowledge and resources.

The NARS project started in 1983 at Peking University. Until now, several working prototypes have been built, in an incremental manner (that is, each with more inference rules and more complicated control mechanism). Currently first-order inference has been finished, and higher-order inference is under developing. Though the whole project is still far from completion, the past experience has shown the feasibility of this approach. For the up-to-date information about the project and latest publication and demonstration, please visit `http://www.cogsci.indiana.edu/farg/peiwang/papers.html`.

References

[Aristotle, 1989] Aristotle (1989). *Prior Analytics.* Hackett Publishing Company, Indianapolis, Indiana. Translated by R. Smith.

[Birnbaum, 1991] Birnbaum, L. (1991). Rigor mortis: a response to Nilsson's "Logic and artificial intelligence". *Artificial Intelligence*, 47:57–77.

[Boddy and Dean, 1994] Boddy, M. and Dean, T. (1994). Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, 67:245–285.

[Carnap, 1950] Carnap, R. (1950). *Logical Foundations of Probability.* The University of Chicago Press, Chicago.

[Chalmers et al., 1992] Chalmers, D., French, R., and Hofstadter, D. (1992). High-level perception, representation, and analogy: a critique of artificial intelligence methodology. *Journal of Experimental and Theoretical Artificial Intelligence*, 4:185–211.

[Dean and Boddy, 1988] Dean, T. and Boddy, M. (1988). An analysis of time-dependent planning. In *Proceedings of AAAI-88*, pages 49–54.

[Dreyfus, 1992] Dreyfus, H. (1992). *What Computers Still Can't Do.* The MIT Press, Cambridge, Massachusetts.

[Elgot-Drapkin et al., 1991] Elgot-Drapkin, J., Miller, M., and Perlis, D. (1991). Memory, reason, and time: the step-logic approach. In Cummins, R. and Pollock, J., editors, *Philosophy and AI*, chapter 4, pages 79–103. The MIT Press, Cambridge, Massachusetts.

[French, 1990] French, R. (1990). Subcognition and the limits of the Turing test. *Mind*, 99:53–65.

[Good, 1983] Good, I. (1983). *Good Thinking: The Foundations of Probability and Its Applications.* University of Minnesota Press, Minneapolis.

[Hempel, 1943] Hempel, C. (1943). A purely syntactical definition of confirmation. *Journal of Symbolic Logic*, 8:122–143.

[Hinton et al., 1986] Hinton, G., McClelland, J., and Rumelhart, D. (1986). Distributed representation. In Rumelhart, D. and McClelland, J., editors, *Parallel Distributed Processing: Exploration in the Microstructure of cognition, Vol. 1, Foundations*, pages 77–109. The MIT Press, Cambridge, Massachusetts.

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

[Hofstadter, 1984] Hofstadter, D. (1984). The copycat project: An experiment in nondeterminism and creative analogies. AI memo, MIT Artificial Intelligence Laboratory.

[Hofstadter, 1985] Hofstadter, D. (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.

[Hofstadter and Mitchell, 1994] Hofstadter, D. and Mitchell, M. (1994). The Copycat project: a model of mental fluidity and analogy-making. In Holyoak, K. and Barnden, J., editors, *Advances in Connectionist and Neural Computation Theory, Volume 2: Analogical Connections*, pages 31–112. Ablex Publishing Corporation, Norwood, New Jersey.

[Holland, 1986] Holland, J. (1986). Escaping brittleness: the possibilities of general purpose learning algorithms applied to parallel rule-based systems. In Michalski, R., Carbonell, J., and Mitchell, T., editors, *Machine Learning: an artificial intelligence approach*, volume II, chapter 20, pages 593–624. Morgan Kaufmann, Los Altos, California.

[Holland et al., 1986] Holland, J., Holyoak, K., Nisbett, R., and Thagard, P. (1986). *Induction*. The MIT Press.

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

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

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

[Medin and Ross, 1992] Medin, D. and Ross, B. (1992). *Cognitive Psychology*. Harcourt Brace Jovanovich, Fort Worth.

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

[Newell and Simon, 1976] Newell, A. and Simon, H. (1976). Computer science as empirical inquiry: symbols and search. The Tenth Turing Lecture. First published in *Communications of the Association for Computing Machinery* 19.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann Publishers, San Mateo, California.

[Penrose, 1994] Penrose, R. (1994). *Shadows of the Mind*. Oxford University Press.

[Piaget, 1960] Piaget, J. (1960). *The Psychology of Intelligence*. Littlefield, Adams & Co., Paterson, New Jersey.

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

[Reeke and Edelman, 1988] Reeke, G. and Edelman, G. (1988). Real brains and artificial intelligence. *Dædalus*, 117(1):143–173.

[Reiter, 1987] Reiter, R. (1987). Nonmonotonic reasoning. *Annual Review of Computer Science*, 2:147–186.

[Rumelhart and McClelland, 1986] Rumelhart, D. and McClelland, J. (1986). PDP models and general issues in cognitive science. In Rumelhart, D. and McClelland, J., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, Foundations*, pages 110–146. The MIT Press, Cambridge, Massachusetts.

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

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

[Shafer, 1976] Shafer, G. (1976). *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, New Jersey.

[Smolensky, 1988] Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11:1–74.

[Strosnider and Paul, 1994] Strosnider, J. and Paul, C. (1994). A structured view of real-time problem solving. *AI Magazine*, 15(2):45–66.

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

[Wang, 1993] Wang, P. (1993). Belief revision in probability theory. In *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*, pages 519–526. Morgan Kaufmann Publishers, San Mateo, California.

[Wang, 1994a] Wang, P. (1994a). A defect in Dempster-Shafer theory. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 560–566. Morgan Kaufmann Publishers, San Mateo, California.

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

[Wang, 1995a] Wang, P. (1995a). Grounded on experience: Semantics for intelligence. Technical Report 96, Center for Research on Concepts and Cognition, Indiana University, Bloomington, Indiana. Available via WWW at *http://www.cogsci.indiana.edu/farg/peiwang/papers.html*.

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

[Wang, 1996] Wang, P. (1996). The interpretation of fuzziness. *IEEE Transactions on Systems, Man, and Cybernetics*, 26(4):321–326.

[Wang, 1999] Wang, P. (1999). A new approach for induction: From a non-axiomatic logical point of view. In Shier, J., Qingyin, L., and Biao, L., editors, *Philosophy, Logic, and Artificial Intelligence*, pages 53–85. Zhongshan University Press.

[Wang, 2001a] Wang, P. (2001a). Abduction in non-axiomatic logic. In *Working Notes of the IJCAI workshop on Abductive Reasoning*, pages 56–63, Seattle, Washington.

[Wang, 2001b] Wang, P. (2001b). Confidence as higher-order uncertainty. In *Proceedings of the Second International Symposium on Imprecise Probabilities and Their Applications*, pages 352–361, Ithaca, New York.

[Wang, 2001c] Wang, P. (2001c). Wason's cards: what is wrong? In *Proceedings of the Third International Conference on Cognitive Science*, pages 371–375, Beijing.

[Zadeh, 1965] Zadeh, L. (1965). Fuzzy sets. *Information and Control*, 8:338–353.