Non-Axiomatic Reasoning System
— Exploring the Essence of Intelligence

Pei Wang

Submitted to the faculty of the Graduate School
in partial fulfillment of the requirements
for the degree
Doctor of Philosophy
in the Department of Computer Science
and the Program of Cognitive Science
Indiana University

August 1995

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements of the degree of Doctor of Philosophy.

Doctoral
Committee

Dr. Douglas R. Hofstadter
(Principal Advisor)

Dr. James T. Townsend

Dr. Gregory J.E. Rawlins

August 30, 1995

Dr. David B. Leake

Dedicated to My Family

Acknowledgments

The research presented in this dissertation was carried out on two beautiful campuses on opposite sides of the earth — Peking University, Beijing, China, and Indiana University, Bloomington, Indiana, USA.

I became interested in artificial intelligence in the early 1980s, when I was an undergraduate student in the Department of Computer Science and Technology at Peking University. At that time AI was a new topic in China, so no faculty members were working on it in the department, and no course was taught on the subject. As a result, I simply read what I could find on AI and related issues. The atmosphere in Peking University in those years was very exciting — just out of the "cultural revolution", the young generation was enthusiastically, seriously, and bravely challenging traditional theories and values in all domains. The influence of the so-called "Spirit of Peking University" was decisive to me; without it, I would not have had the confidence to work on my own ideas for more than 10 years without major signs of acceptance from the AI community.

As I had very limited access to established researchers and reference materials, many of my ideas were inspired by discussions with my friends. Among them I want to mention Chen Gang, Cai Shan, Sun Yongping, and in particular, Bai Shuo (names written in the traditional Chinese order, with the family name followed by the given name).

After developing some preliminary ideas in my thesis for my bachelor's degree, I decided to pursue them further when I became a graduate student in the same department. Professor Xu Zhuoqun found my ideas promising, and agreed to be my advisor. Without his support, it would have been difficult for me to finish the earliest version of my system in three years. In my master's thesis, finished in 1986, almost all the ideas presented in this Ph.D dissertation were proposed, though they were not developed nearly as fully there.

After I got my master's degree in July 1986, I was offered the position of Lecturer by the department. I accepted it, because Peking University was so much like a home for me. During the summer vacation that year, a classmate, Yan Yong, persuaded me to join the translation project of Douglas Hofstadter's *Gödel, Escher, Bach: an Eternal Golden Braid*. The translation process turned out to be very exciting, and I found many of my own ideas were presented in the book in a beautiful way. As a consequence, I wrote to Professor Hofstadter and then he and I exchanged some research papers, at which point both of us noticed a surprising degree of overlap in our research projects. Consequently, he arranged for me to join his research group at Indiana University, and I was also accepted by the Ph.D program in computer science and cognitive science at IU.

This time I was lucky again to have an open-minded advisor. Though Professor Hofstadter did not share all my opinions about intelligence, he still provided me the support I needed to continue my research. His opinions have always been stimulating to my thought. Thanks to his help, the four years I have spent on the peaceful Bloomington campus have been very productive and pleasant. His detailed corrections and comments on several drafts have greatly improved the quality of this dissertation.

The other members of the Center for Research on Concepts and Cognition were very helpful. They include David Moser, Robert French, Terry Jones, David Chalmers, Gary McGraw, Jim Marshall, and John Rehling. Helga Keller gave me enormous help in administrative affairs. Jeff Logan played a major role in the improvement of my English. Carol Hofstadter's kindness to my family is unforgettable for us. I also want to thank the other members of my advisory committee — David Leake, Gregory Rawlins, and James Townsend — for their valuable comments and suggestions.

In the current implementation of the project, some source code from the book *Object-Oriented Programming with C++ and OSF/Motif* (Prentice Hall, 1992. ISBN 0-13-630252-1), by Douglas Young, has been borrowed.

I dedicate this dissertation to my family. My parents, Wang Zhizhong and Zhai Minghui, have always given me much encouragement. I hope this dissertation will at least partially satisfy their expectations of me. My son, Wang Siyue, always wants to know what I am doing in my "research", but I have not yet found a way to explain it to him clearly. And finally, my wife, Sun Hongyuan, is always understanding of my pursuits, and she has contributed and sacrificed in many ways in order to further my work. I hope I have not wasted or misused her efforts.

Abstract

Every artificial-intelligence research project needs a working definition of "intelligence", on which the deepest goals and assumptions of the research are based. In the project described in the following chapters, "intelligence" is defined as the capacity to adapt under insufficient knowledge and resources. Concretely, an intelligent system should be finite and open, and should work in real time.

If these criteria are used in the design of a reasoning system, the result is NARS, a non-axiomatic reasoning system.

NARS uses a term-oriented formal language, characterized by the use of subject–predicate sentences. The language has an experience-grounded semantics, according to which the truth value of a judgment is determined by previous experience, and the meaning of a term is determined by its relations with other terms. Several different types of uncertainty, such as randomness, fuzziness, and ignorance, can be represented in the language in a single way.

The inference rules of NARS are based on three inheritance relations between terms. With different combinations of premises, revision, deduction, induction, abduction, exemplification, comparison, and analogy can all be carried out in a uniform format, the major difference between these types of inference being that different functions are used to calculate the truth value of the conclusion from the truth values of the premises.

Since it has insufficient space–time resources, the system needs to distribute them among its tasks very carefully, and to dynamically adjust the distribution as the situation changes. This leads to a “controlled concurrency” control mechanism, and a “bag-based” memory organization.

A recent implementation of the NARS model, with examples, is discussed. The system has many interesting properties that are shared by human cognition, but are absent from conventional computational models of reasoning.

This research sheds light on several notions in artificial intelligence and cognitive science, including symbol-grounding, induction, categorization, logic, and computation. These are discussed to show the implications of the new theory of intelligence.

Finally, the major results of the research are summarized, a preliminary evaluation of the working definition of intelligence is given, and the limitations and future extensions of the research are discussed.

Contents

Acknowledgments **v**

Abstract **viii**

1 Introduction **1**

2 Theoretical Foundations **4**

2.1 AI paradigms 4

2.2 Intelligence: a working definition 13

2.3 Reasoning systems 18

3 Language **22**

3.1 Experience-grounded semantics 22

3.2 A binary inheritance relation 29

3.3 Uncertainty 34

3.4 Experience: ideal vs. real 46

3.5 Grammar 49

4 Inference Rules 51

4.1 Revision and choice 51

4.2 Syllogisms 59

4.3 Rules for the other two inheritance relations 68

4.4 A summary of the rules 71

5 Control Mechanisms 78

5.1 Controlled concurrency 79

5.2 Bags and chunks 86

5.3 An atomic step 89

5.4 Priority and durability 91

6 Computer System 96

6.1 Internal structure 96

6.2 User interface 98

6.3 Examples 105

6.4 Performance 116

7 Theoretical Implications 119

7.1 Symbol-grounding 119

7.2 Induction 124

7.3 Categorization 127

7.4 Is NARS a logic? 131

7.5 NARS as an inheritance network 136

7.6 Is it still computation? 139

8 Conclusion 147

8.1 Major results 147

8.2 Evaluation 149

8.3 Limitations and extensions 151

List of Tables

3.1 Relations among uncertainty measurements. 44

3.2 Grammar of NAL2. 50

4.1 Revision rules. 71

4.2 Syllogistic rules. 72

4.3 Truth-value functions. 73

4.4 Backward inference rules. 76

List of Figures

6.1 Screen snapshot of NARS 3.0. 99

6.2 Main window. 100

6.3 Input window. 101

6.4 Demonstration window. 104

6.5 Bag contents window. 104

6.6 Parameter window. 105

7.1 An inheritance network. 137

7.2 Rules of the inheritance network. 137

1

Introduction

The research presented in this dissertation is an attempt to explore the essence of *intelligence*, and to reproduce it in *artificial* entities — namely, computer systems.

To achieve such an *artificial intelligence* (AI), research needs to be carried out on three levels of abstraction.

First, there are many philosophical and methodological questions that need to be considered. What is the basic difference between systems that have intelligence and those that lack intelligence? What is the basic difference between artificial intelligence and natural intelligence? What is the most fruitful way to study artificial intelligence? How to evaluate the paradigms that have played major roles in the history of AI? What are the relationships between AI and related disciplines? My opinions on these questions constitute the foundation for my whole research project, by providing a clear aim and a set of principles. The research results on this first level constitute a *theory* of intelligence in general.

Second, guided by the theory, a formal model is designed to develop the theory in a precise manner. By "formal model", I mean that the details of the model are described symbolically or mathematically. All ambiguities in the theory need to be clarified on this level. When simplifications become inevitable, they should follow the principles developed in the theory. The research results on this second level constitute a medium-independent *model* of intelligence.

Finally, the model is implemented on a computer system. All the mechanisms described in the model should be implanted into programs, and these programs should be runnable in a currently available software and hardware environment. The results on this final level constitute an intelligent computer *system*.

The organization of this dissertation is based on these three levels.

In Chapter 2, the philosophical and methodological foundations of the project are developed. The major conclusions are that intelligence is characterized by the ability to adapt to the environment under insufficient knowledge and resources, and that a reasoning system is a suitable platform for the study of artificial intelligence.

Chapter 3, 4, and 5 present a formal model of reasoning that is intelligent in the above-defined sense.

Chapter 3 describes the formal language used in the model, which has a term-oriented syntax and experience-grounded semantics, and can represent several types of uncertainty.

Chapter 4 discusses the inference rules of the model, such as revision, choice, deduction, induction, abduction, exemplification, comparison, analogy, and backward inference. All of them are carried out in similar formats.

Chapter 5 introduces the control mechanism of the model, including the concepts of "controlled concurrency" and "bag-based memory organization".

A computer implementation of the model is briefly presented in Chapter 6. The major design issues are discussed. Several concrete examples are analyzed. Finally, the performance of the system is evaluated.

Chapter 7 returns to the theoretical level. In the light of the model and its implementation, several important problems in artificial intelligence and cognitive science are revisited.

Finally, in Chapter 8 the major results of the research are summarized, a preliminary evaluation of the working definition of intelligence is given, and the limitations and future extensions of the research are discussed.

Some of the material presented in this dissertation is adapted from my previous publications and technical reports, which are listed in the bibliography and are cited at relevant places in the following chapters.

2

Theoretical Foundations

2.1 AI paradigms

Attempts to clarify the notion of "intelligence" and to explore ways to realize it in computing machinery can be traced back to (Turing, 1950), in which Turing suggested passing an imitation test as a sufficient condition for "being intelligent".

The debate on the essence of intelligence has now been going on for decades, and there is still little sign of consensus (Kirsh, 1991). As a matter of fact, almost everyone in the field has unique personal opinions about how the word "intelligence" should be used, and these opinions in turn influence the choice of research goals and methods, as well as serve as standards for judging other researchers' projects.

Though it is too early to look for a general definition, the choice of a *working definition* of intelligence is inevitable and crucial for an AI researcher, because it determines the foundation for the whole research effort (Wang, 1994d). In my case, I seek a working definition of intelligence that satisfies the following requirements:[1]

[1] These criteria, under the names of *similarity*, *exactness*, *fruitfulness*, and *simplicity*, were suggested by Carnap in his attempt to clarify the concept of "probability" (Carnap, 1950).

Faithfulness. Though “intelligence” has no precise 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.

Sharpness. 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. 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.

With these criteria in mind, we can evaluate current AI paradigms by analyzing their working definitions of intelligence. Obviously, no working definition of intelligence can be perfect according to the above requirements, but that does not mean that all of them are equally good. Different working definitions lead AI research down different pathways, and some of those directions, although possibly fruitful for some other purposes, make little contribution to the study of intelligence (Wang, 1994d).

Since it is impossible to study each of the existing working definitions of intelligence one by one (there are too many of them), we will group them into several categories. As usual, a definition may belong to more than one category at the same time.

Generally speaking, work towards realizing artificial intelligence has two major motivations. As researchers in a field of science, we want to learn how the human mind, and "mind" in general, works; and as developers of 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 are they "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 five categories, to see where these working definitions of intelligence will lead AI.

- **To simulate the human brain**

Intelligence is an emergent outcome of the human brain, so maybe AI should attempt to simulate a brain in a computer system as faithfully as possible. This philosophy reaches its extreme form in the work of 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 a *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 may be helpful for the building of the latter.

- **To duplicate human behavior**

Given that we always judge the intelligence of other people by their behavior, it is natural to use "reproducing the behavior caused by the human brain as accurately as possible" as the aim of AI. In this way, we can draw "a fairly sharp line between the physical and the intellectual capacities of a man" (Turing, 1950).

Such a working definition of intelligence asks researchers to use "passing the Turing test" as a sufficient *and necessary* condition for having intelligence, and to take psychological evidence seriously, as Newell and company do in the Soar project (Newell, 1990).

This approach can be criticized from various angles:

Is it sufficient? Searle argues that even if a computer system can pass the Turing test, it still cannot *think*, because it lacks the *causal capacity* of the brain to produce *intentionality*, which is a biological phenomenon (Searle, 1980). However, he does not demonstrate convincingly why thinking, intentionality, and intelligence cannot have a high-level (higher than the biological level) description.

Is it possible? 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.

Is it necessary? 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). As a working definition for intelligence, 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. Furthermore, we would have to say that no other animal but a human had vision, if we defined "vision" as "indistinguishable from a human in terms of reactions to light stimuli to the eye" or something like that. This strikes me as a very unnatural and unfruitful way to use concepts.

In summary, 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". Actually, Turing does not claim that passing the imitation test is a necessary condition for being intelligent. He just thinks that if a machine can play the game satisfactorily, we need not be troubled by the question (Turing, 1950).

- **To solve hard problems**

In everyday language, "intelligent" is usually applied to people who can solve hard problems. Many definitions of intelligence come from this usage. 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." Usually, such systems are developed by analyzing domain knowledge and experts' strategies, then building them into a computer system.

This movement has drawn in many researchers, produced many practically useful systems, attracted significant funding, and thus made important contributions to the development of the AI enterprise. 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 can, or substitute a word in a file as fast as an editing program can. If we say "hard for computers", then AI becomes "whatever hasn't been done yet", which has been dubbed "Tesler's Theorem" (Hofstadter, 1979) and the "gee whiz view" (Schank, 1991).

The view that AI is a "perpetually expanding 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, independently of the brain mechanisms that carry it out and the practical domains that it can be applied to. The direct aim of this kind of study is to build a computer system with the desired function(s).

This approach has produced, and will continue to produce, information-processing 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 the following weaknesses:

1. 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; Hofstadter and FARG, 1995).

2. Having any particular cognitive function is not enough to make a system intelligent. For example, problem-solving by exhaustive search is usually not considered intelligence, and many unintelligent animals have excellent perceptual capacities.

3. 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. For example, we can grow a tree, but we cannot generate a leaf *alone*, although a leaf is much simpler than a tree. There is considerable evidence to suggest that intelligence is such a phenomenon. As Piaget said: "Intelligence in action is, in effect, irreducible to everything that is not itself and, moreover, it appears as a total system of which one cannot conceive one part without bringing in all of it." (Piaget, 1963)

- **To develop new principles**

In summary, the *structure* approach contributes to neuroscience by building brain models, the *performance* 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.

The research presented in this dissertation is based on the premise that what distinguishes intelligent from unintelligent systems is their basic *principles* of information processing. I will develop this premise in the following pages.

2.2 Intelligence: a working definition

Here is my working definition of intelligence:

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

An *information-processing system* is a system whose internal activities and interactions with its environment can be studied *abstractly* — that is, without specifying the physical hardware that carries out the activities and interactions. Usually, such a system has certain *tasks* (given by the environment, or generated by the system itself) to carry out. To do this, the system takes various *actions*, guided by its *knowledge* about how the actions and the tasks are related. Any internal activity costs the system some *resources*. According to this definition, all human beings and computer systems, as well as many animals and automatic control systems, can be described as information-processing systems.

The *environment* of such a system may be the physical world (if the system has sensory–motor capacities), 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, respectively. For the system, recognizable patterns of input and producible patterns of output constitute its *interface language*.

To *adapt* means that the system learns from its experiences. It carries out tasks and adjusts its internal structure to improve its resource efficiency, under the assumption that future situations will be similar to past situations. Not all information-processing systems adapt to their environment. For instance, a traditional computing system gets all of its knowledge during its design phase (or before its "birth"). After that, its experience contains tasks only, and the results do not further contribute to the experience of the system. Indeed, to acquire new knowledge, such a system would have to be redesigned, which certainly cannot be done by communicating with a human user in its interface language. On the other hand, not all experience-related changes can be called "adaptation". Adaptation takes place only if the change involved helps to make the system work *better*, provided that the environment is relatively stable.

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 constraints attached to them.

Open: No constraints are put on the contents of 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. Such a system acts, for all intents and purposes, as if its knowledge and resources were indeed sufficient.

Not all information-processing 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 machines) 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, 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:

- 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 constraint, 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;
  etc., etc.

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). This assertion is consistent with the usages of the two words in natural language: we are willing to call many animals, computer systems, and automatic control systems "adaptive", but not "intelligent".

To be sure, what has been proposed in my definition 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.

We can also find similar attempts to base intelligence on certain basic principles (such as some form of rationality) in the following ideas:

> *Type II rationality* (Good, 1983): "Type II rationality is defined as the recommendation to maximize expected utility allowing for the cost of theorizing. It involves the recognition that judgments can be revised, leading at best to consistency of *mature* judgments."
>
> *Bounded rationality* (Simon, 1983): "Within the behavioral model of bounded rationality, one doesn't have to make choices that are infinitely

> deep in time, that encompass the whole range of human values, and in which each problem is interconnected with all the other problems in the world."
>
> *Minimal rationality* (Cherniak, 1986): "We are in the finitary predicament of having fixed limits on our cognitive resources, in particular, on memory capacity and computing time."
>
> *Limited rationality* (Russell and Wefald, 1991): "Intelligence was intimately linked to the ability to succeed as far as possible given one's limited computational and informational resources."

Medin and Ross also put it quite clearly: "Much of intelligent behavior can be understood in terms of strategies for coping with too little information and too many possibilities." (Medin and Ross, 1992)

Given all this, what is *new* in my approach? I claim that 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 clarification of the phrase "with insufficient knowledge and resources" as meaning *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.

How is my working definition of intelligence different from the others discussed in the previous section? In the following chapters, we can see that a system developed on such a foundation has 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[2], 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 *performance*. 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.

2.3 Reasoning systems

The preceding ideas about intelligence can be applied to various types of information systems, such as perception systems, planning systems, and so on. However, I have chosen to develop a *reasoning system* as the concrete platform of my research. A reasoning system, in a broad sense, is an information-processing system that has the following components:

1. *a formal declarative language*, defined by a grammar, for communication between the system and its environment (a human user or another computer, not the physical world), and for the internal representation of the system;

[2] Actually, *hard problems* are those for which a solver (human or computer) has insufficient knowledge and resources. This implies that the common property shared by AI problems is the set of *conditions* under which the problems need to be solved.

2. *a semantics* of the formal language that determines the meanings of the words and the truth values of the sentences in the language;

3. *a set of inference rules* that is defined formally, and that can be used to match questions with knowledge, to infer conclusions from premises, to derive subquestions from questions, and so on;

4. *a memory* that systematically stores both questions and knowledge, and provides a work place for inferences;

5. *a control mechanism* that is responsible for resource management, for choosing premises and inference rules in each step of inference, and for processing space requirements.

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.

Being a reasoning system is neither necessary nor sufficient for being intelligent, but an *intelligent reasoning system* provides a suitable object for the study of intelligence for the following reasons:

1. 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.

2. 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.

3. Most research on reasoning systems is carried out within a paradigm based on premises directly opposed to mine 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 constraints 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 research directions in 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, has been chosen as the name for the intelligent reasoning system to be introduced in the following chapters.

3

Language

In this chapter, the formal language used in the NARS model, along with its grammar and semantics, is described and related to the theory presented in the previous chapter. This language is used by NARS for both internal representation and external communication.

3.1 Experience-grounded semantics

Semantics is the study of how the items in a language are related to the environment in which the language is used. Concretely, semantics is the theory of *meaning* and *truth*. By asking questions like "What is the meaning of a term?" and "What is the truth value of a sentence?", we are looking for the *principles* that determine meaning and truth in general, rather than the meaning of a specific word or the truth of a specific sentence.

A computerized reasoning system often uses an artificial language. The syntax of the language is usually precisely defined by a formal grammar. The system carries out inferences in the language according to some formal inference rules. For such a system, we need a semantics for two major reasons. When designing the system, we need to choose inference rules that will lead to desired conclusions; when communicating with the system, we need to understand the system's language.

Model-theoretic semantics, together with some variants, is the dominant paradigm in the semantics of formal languages.

Formal languages were developed in the study of the foundation of mathematics by Leibniz, Frege, Russell, Hilbert, and so on. A basic motivation for formal languages is to get rid of the ambiguities in natural language so that an objective and accurate artificial language can be created. Model-theoretic semantics was founded by Tarski. Although his primary target was formal language, he also hoped that the ideas could be applied to reform everyday language (Tarski, 1944). This approach characterizes the "logic-based" branch of AI (McCarthy, 1988; Nilsson, 1991).

For a language **L**, defined by a finite formal grammar, a model **M** consists of the relevant part of some domain, which can be described in another language **ML**, and an interpretation **I**, which maps the items in **L** onto the objects in the domain, labeled by words in **ML**. **ML** is referred to as a "meta-language", which can be either a natural language, like English, or another formal language.

Given the above components, the *meaning* of a term in **L** is defined as its image in **M** under **I**, and whether a sentence in **L** is *true* is determined by whether it is mapped by **I** onto a "state of affairs" that holds in **M**. For a reasoning system, valid inference rules are those that always derive true conclusions from true premises.

According to this view, as Tarski put it, "semantics is a discipline which deals with certain relations between expressions of a language and the objects 'referred to' by those expressions." (Tarski, 1944)

Let us see what is implied by the above definitions. According to model-theoretic semantics, for any formal language, the necessary and sufficient condition for its terms to have meanings and for its sentences to have truth values is the existence of a model. In different models, the meanings and truth values in the language may change; however, these changes are not caused by *using* the language. A reasoning system **R** that works on representations expressed in **L** does not depend in any way on the semantics of **L**. That means, on the one hand, that **R** has no *access* to the meanings of terms and the truth values of sentences — it can distinguish terms only by their forms, and can derive new sentences from old ones only according to its inference rules, but does not put any constraints on how the language can be interpreted. On the other hand, what **R** does to the terms and sentences in **L** has *no influence* on their meanings and truth values. When working *within* such a system, as Russell said, "we never know what we are talking about, nor whether what we are saying is true" (Russell, 1901).

Such properties are useful for meta-mathematics, where abstract patterns of ideal inference are studied, and the patterns can be applied to different domains by constructing different models. And thus, the study of semantics has contributed significantly to the development of meta-mathematics. As Tarski said, "As regards the applicability of semantics to mathematical science and their methodology, i.e., to meta-mathematics, we are in a much more favorable position than in the case of empirical sciences." (Tarski, 1944)

However, the attempt to apply this idea to the semantics of natural language runs up against many problems (Ellis, 1993; Lakoff, 1994; Palmer, 1981). It seems that natural language is too subtle and fluid to be put into the frame of model-theoretic semantics. Also, this approach works poorly for non-deductive inferences (Birnbaum, 1991; McDermott, 1987), despite various attempts to render the theory more flexible by introducing ideas like possible worlds and multi-valued propositions (Carnap, 1950;

Halpern, 1990; Kyburg, 1992; Zadeh, 1986).

The problems with model-theoretic semantics are often used as arguments against so-called "strong AI". For example, Searle's assertion in his "Chinese room" argument that "computers are syntactic, but the human mind is semantic" (Searle, 1980) is directly based on the assumption that all computerized symbol manipulations are intrinsically wedded to model-theoretic semantics, so that uninterpreted symbols are by definition meaningless.

Model-theoretic semantics has been criticized by many authors for its rigidity (Birnbaum, 1991; McDermott, 1987). However, without a powerful competitor, the solution is far from clear. As McDermott said: "The notation we use must be understandable to those using it and reading it; so it must have a semantics; so it must have a Tarskian semantics, because there is no other candidate." (McDermott, 1987) Some people believe that it is the very idea of formalizing the language and the inference rules that should be abandoned. They try other approaches, such as neural networks and robots, in the hope that these will ground meaning and truth in perception and action (Birnbaum, 1991; Harnad, 1990).

What is the fundamental difference between natural and artificial (formal) languages? Why must the latter be rigid, fixed, determined, and unambiguous? This question is especially important for artificial intelligence, because here we want a computer system either to use a natural language directly or to use an artificial language in a more fluid and flexible way.

Some researchers suggest that the reasoning system itself (human or computer), rather than the world it deals with, should be used as the domain of the language the system uses. Thus, one could posit that the meaning of a particular term is a particular "concept" that the system has, and the truth value of a sentence is the system's "degree of belief" in that sentence. This idea sounds reasonable, but it does not answer the original question: how are "concepts" and "degrees of belief" dependent upon the outside world? Without an answer to that question, such a solution "simply pushes the problem of external significance from expressions to ideas" (Barwise and Perry, 1983).

In NARS, I explore another possibility: the abandonment of model-theoretic semantics in favor of another type of semantics for an intelligent reasoning system, yet one that still uses a formal language and formal inference rules.

The model-theoretic approach, in asserting the existence of a model **M**, presumes that there is, at least in principle, a consistent, complete, and static description of (the relevant part of) the environment in a language **ML**, and that such a description, a "state of affairs", is at least partially known, so that the truth value of some sentences in **L** can be determined accordingly. These sentences then can be used as premises for all inferences. It is also required that all valid inference rules be truth-preserving, which implies that only true conclusions (no matter how expensive they are) are attainable.

Such conditions hold only when a system has *sufficient knowledge and resources* with respect to the problems to be solved. "Sufficient knowledge" means that the desired results can be obtained by inference from available knowledge alone, so no additional knowledge will be necessary; "sufficient resources" means that the system can afford the time–space expense of the inference, so no approximation is necessary. These are exactly the assumptions we accept when working *within an axiomatic system*. Therefore it is no surprise that model-theoretic semantics works fine there.

Now let us consider the opposite situation: a system that works with *insufficient knowledge and resources*. Model-theoretic semantics cannot be applied in such a situation for the following reason. If we still define truth as "agreement with reality", so that truth values cannot be threatened by the acquisition of new knowledge, then no sentence can ever be assigned a truth value under the above assumptions, because in an open system, all knowledge can (by definition) be challenged by future evidence. Model-theoretic semantics also prohibits the system from using fluid or fuzzy concepts or generating new concepts, since in general there is no way to confirm that these concepts really correspond to objects that "exist in the domain".

However, it does not follow that in such a situation semantic notions like "truth" and "meaning" are meaningless — after all, if that were the case, then we could not define truth and meaning in any realm beyond mathematics. For our current purpose, we need a different type of semantics.

As was stated earlier, semantics studies how the items in a language are related to the environment in which the language is used. With insufficient knowledge and resources, what relates the language **L**, used by a system **R**, to the environment is not a *model*, but the system's *experience*, which consists of the knowledge and tasks provided by the environment to the system during their interaction. For a reasoning system like NARS, the experience of the system is a stream of sentences in **L**, provided by a human user or another computer.

In such a situation, the basic semantic notions of "meaning" and "truth" still make sense. The system may treat terms and sentences in **L**, not solely according to their syntax (shape), but in addition taking into account their relations to the environment.

To a human designer or user, a semantics is necessary because we want to make the system *adaptive* — that is, we want it to behave according to its experience. To this end, the system needs to be able to judge the truth values of sentences according to whether, or how much, they are *supported* by its experience, and to distinguish the meaning of terms according to their *relations* in its experience.

In summary, for an adaptive system working with insufficient knowledge and resources, model-theoretic semantics is no longer applicable. What we need is what I shall henceforth refer to as *experience-grounded semantics*.

As descriptions of an environment, what is the difference between “model” and “experience”? The following are the main differences:

1. A model is a complete description of an environment, whereas experience is only a partial description of it.

2. A model must be consistent, whereas pieces of knowledge in experience may conflict with one another.

3. A model is static, whereas experience stretches out over time.

4. A model of **L** is represented in another language **ML**, and is not necessarily accessible to a system that uses **L**, whereas experience is represented in **L** itself, and is accessible to the system.

Obviously, NARS should not (and cannot) use “true” and “false” as the only truth values of sentences. To handle conflicts in experience properly, we need to determine what counts as positive evidence in support of a sentence, and what counts as negative evidence against it, and in addition we need some way to measure the *amount* of evidence in terms of some fixed unit. In this way, a truth value will simply be a numerical summary of relevant evidence.

However, as was mentioned above, “evidence” in NARS is represented in **L**, too. Therefore, the truth value of a sentence in **L** is defined by a set of sentences, also in **L**, with their own truth values — which seems to have led us into a circular definition or an infinite regress.

The way out of this seeming circularity in NARS is "bootstrapping" — taking a small subset of **L** to define the truth values of sentences and meanings of terms in **L** in general.

3.2 A binary inheritance relation

In the research presented in this dissertation, I take a path that is opposite to the usually accepted one. Instead of first defining a formal language, then attaching a semantics to it, I introduce the desired semantics first (guided by my working definition of intelligence), then look for a formal language that can support such a semantics. The advantage of such an approach is argued in (Ellis, 1993).

From the previous discussion, we can see that what NARS needs is a formal language in which the meaning of a term is represented by its relationship with other terms, and the truth value of a sentence is determined by available evidence. For these purposes, the concept of (positive or negative) evidence should be naturally introduced into the language.

Unfortunately, the most popular formal language used in first-order predicate logic does not satisfy the requirement. As revealed by (Hempel, 1943), the concept of "confirmation", or positive evidence, cannot be easily defined in the first-order (predicate-oriented) language. Let us suppose that "Ravens are black" is formulated as $(\forall x)(Raven(x) \rightarrow Black(x))$, and that a piece of confirming (positive) evidence is defined as a constant that when substituted for the variable x makes both the condition and the conclusion true. Such a treatment looks natural, since accordingly a black raven is referred to as a piece of positive evidence for the sentence, while a white raven is a piece of negative evidence against it.[1] However, a green shirt will

[1] There is a subtle difference between "Ravens are black" and "All ravens are black". The former

also be counted as a piece of positive evidence for the sentence, because it confirms the "logically equivalent" sentence $(\forall x)(\neg \mathit{Black}(x) \rightarrow \neg \mathit{Raven}(x))$. Such a result is highly counterintuitive, and may cause many problems (for example, a green shirt is also a piece of positive evidence for "Ravens are white", for exactly the same reason).

Here I will not discuss the various solutions proposed for this paradox. Almost all these attempts are still within the framework of first-order predicate logic, whereas in my own approach, I find that it makes more sense simply to give up that implicit constraint, and to switch to another type of language.

A traditional rival to predicate/propositional logic is known as *term logic*. Such logics, exemplified by Aristotle's logic, have the following features (Bocheński, 1970; Englebretsen, 1981):

1. Each sentence consists of a *subject term* and a *predicate term*, which are related by a *copula*.
2. The copula is intuitively interpreted as "to be".
3. An inference rule takes two sentences that share a common term as premises, and from them derives a conclusion in which the other two (unshared) terms are related by a copula.

Term logic appeared earlier than predicate logic, and it was the major paradigm until the rise of mathematical logic (Bocheński, 1970). From then on, it was generally treated as a limited and obsolete idea. But this was too negative a judgment. Though predicate logics work well in axiomatic systems, we will see, in the following chapters, that term logics are in fact more suitable for non-axiomatic systems.

is a general statement, but not a universal claim like the latter. Whereas a counterexample will totally refute the latter, it simply makes the former less probable.

In the simplest type of term logic, there is only one type of copula, and all the terms are "atomic" — that is, have no internal structure. In this way, we get an *Inheritance Logic* (IL) (Wang, 1994c).

Definition 1 *A* term *is a string of letters in an alphabet.*

Definition 2 *The* binary inheritance relation*, "$\sqsubset$", is a reflexive and transitive relation between two terms.*

(The reason for this terminology will be explained toward the end of this section.)

Definition 3 *A* statement *consists of two terms related by the inheritance relation. In the statement "$S \sqsubset P$", S is the* subject term *and P is the* predicate term*.*

Definition 4 **L0** *is a formal language whose sentences are statements defined above.*

The intuitive meaning of the binary inheritance relation is closely related to many well-known relations — for instance, "IS-A" (in semantic networks), "belongs to" (in Aristotle's syllogisms), "subset" (in set theory), "inheritance assertion" (in inheritance systems (Touretzky, 1986)), as well as many relations studied in AI, psychology, and philosophy, such as "type–token", "category–instance", "general–specific", and "superordinate–subordinate" (Brachman, 1983). What makes it different from the others is: it is a relation between two *terms*, and the relation is completely defined by the two properties: *reflexivity* and *transitivity*.

We will define exactly two inference rules in IL, corresponding to reflexivity and transitivity, respectively. With these two rules, from any non-empty and finite set of statements K (as premises), the following algorithm can generate the set of all conclusions K^*:

1. Initially let $K^* = K$;
2. For each term T appearing in K^*, put "$T \sqsubset T$" into K^* (if it is not already there);
3. For each pair of statements "$S \sqsubset M$" and "$M \sqsubset P$" in K^*, put "$S \sqsubset P$" in K^* (if it is not already there).

The last two steps need to be iterated over and over again until all possibilities have been exhausted.

To clarify how a particular term T is related to other terms, the *extension* and *intension* of T, relative to a set of statements K, will be defined in the following manner:

Definition 5 *The* extension *of a term T, E_T, is the set of terms x such that "$x \sqsubset T$" is in K^*. The* intension *of a term T, I_T, is the set of terms x such that "$T \sqsubset x$" is in K^*.*[2]

This definition and a modified version of it, Definition 12 (which is presented at the very end of the next section), are of great importance to NARS. Traditionally, *extension* and *intension* refer to two quite different aspects of the meaning of a term: roughly speaking, its *instances* and its *properties*. A term's extension is usually defined as that set of *objects* in a "physical world" that are denoted by the given term; the term's intension is usually defined as a *concept* in a "Platonic world" which denotes or describes the given term (Bocheński, 1970; Inhelder and Piaget, 1969). In spite of minor differences among the exact ways the two words are used by different authors, they always indicate relations between a term in a language and something *outside*

[2] The extension and intension of a term are sets, but a term itself is not a set.

the language. By contrast, in the current theory they are defined using (the two sides of) a binary relation between two terms, which is *within* the language, yet even so, the definition retains the intuitive feature that "extension" refers to instances, in a sense, and "intension" refers to properties.

The definitions have the following implications:

1. "Extension" and "intension" are defined in a symmetric way, so that for any result about one of them, there is a dual result about the other.

2. Each statement in the system's experience reveals part of the intension for the subject term and part of the extension for the predicate term.

3. Since the inheritance relation is reflexive, any given term has a non-empty extension and a non-empty intension — both of them contain at least the term itself.

From the definition, it is not difficult to get the following two results.

First, "$S \sqsubset P$" holds if and only if S's extension is fully contained in P's extension, and also if and only if P's intension is fully contained in S's intension. In other words, the statement "There is an inheritance relation from S to P" is equivalent to either "P inherits S's extension" or "S inherits P's intension". This is the reason that "$\sqsubset$" is called an *inheritance relation.* Intuitively, such a relation indicates that one term *can be used as*, or *inherits the relations of*, the other in a certain way (see (Hofstadter, 1995) for a discussion of "as"). If a system knows "$S \sqsubset P$", then S can serve in place of P in sentences of the form "$P \sqsubset x$", and P can serve in place of S in sentences of the form "$x \sqsubset S$", where x is an arbitrary term. Conversely, if all x's that satisfy "$x \sqsubset S$" also satisfy "$x \sqsubset P$", or all x's that satisfy "$P \sqsubset x$" also satisfy "$S \sqsubset x$", we have "$S \sqsubset P$" (Wang, 1994c).

Second, the *extensions* of S and P precisely coincide if and only if their *intensions* precisely coincide. This means that the extension and intension of a term are mutually determined. Therefore, given one of them, the other can be uniquely obtained.

Definition 6 *Given a set of statements K as the experience of the system, a statement is* true *if it belongs to K^*, otherwise it is* false*. The* meaning *of a term consists of its extension and intension.*

In this way, truth and meaning are defined in terms of the experience of a system.

Later, we will also see that K^* corresponds to the set of all available *evidence* to the system, which is contained in, or derived from, the system's experience.

Now we have finished our description of a simple term logic IL, which has a term-oriented grammar, experience-grounded semantics, and inheritance-based inference rules. However, in designing this system, we did not take the insufficiency of knowledge and resources into consideration. As a result, we still have binary truth values for the statements in IL, and constant meanings for the terms.

3.3 Uncertainty

Since we are assuming that NARS has insufficient knowledge, we know that the above results are not practical, because we did not balance positive and negative evidence, nor did we take into account the influence of future evidence. To do these, a binary truth value is not enough — we need more information about evidence.

The sentences that actually appear in the experience and knowledge base of NARS all have the form "$S \subset P$", where "$\subset$" is the "basic inheritance relation", which is a generalization of the "$\sqsubset$" relation. "$S \subset P$" is not simply true or false — it is confirmed and/or refuted to various degrees by available evidence (from the system's experience).

As was stated at the end of Section 3.1, NARS cannot ground the truth values of all of its sentences simply by taking other sentences in the same language as pieces of evidence — that would lead to circularity and empty, runaway feedback loops. Instead, we will use the language introduced in the previous section, with its binary truth values, to construct a hypothetical body of "ideal experience", and will interpret any "$S \subset P$" sentence, including its (non-binary) truth value, *as if* there were hidden binary "$x \sqsubset y$" sentences behind the scenes, playing the role of pieces of evidence. Indeed, the sole purpose of defining "$\sqsubset$" (keep in mind that it never appears in NARS' actual experience) is to help us in defining the notion of *evidence* for sentences using "$\subset$".

Concretely, we define evidence for a given non-binary "$\subset$" sentence in terms of two binary "$\sqsubset$" sentences (with respect to a given K).

Definition 7 *A piece of* positive *evidence (with a unit weight) for "$S \subset P$" is a term M such that both "$M \sqsubset S$" and "$M \sqsubset P$" are in K^*, or both "$P \sqsubset M$" and "$S \sqsubset M$" are in K^*. A piece of* negative *evidence (with a unit weight) for "$S \subset P$" is a term M such that "$M \sqsubset S$" is in K^* but "$M \sqsubset P$" is not, or "$P \sqsubset M$" is in K^* but "$S \sqsubset M$" is not.*

The intuition behind the above definition is as follows: like "$S \sqsubset P$" in the previous section, "$S \subset P$" also states "S inherits the intension of P, and P inherits the extension of S". Therefore, if M is in the extension (or intension) of both S and P, it is a piece of positive evidence for the statement; if M is in the extension of S (the intension of P), but not in the extension of P (the intension of S), it is a piece of negative evidence for the statement. K^*, defined in the previous section, corresponds to a hypothetical body of purely binary experience of the system.

Hempel's paradox does not arise here, because a green shirt counts as neither positive nor negative evidence for "Ravens are black", according to the previous definition. Furthermore, the negation of a term is not a term in NARS 3.0, so there simply is no such sentence as "Non-black-things are non-ravens" — that is, there is no contrapositive sentence that is "logically equivalent" to "Ravens are black".[3]

In the following, let us call a sentence with an attached truth value a *judgment*. The truth value is determined by the system's experience. Given a system with experience K, the truth value of a judgment "$S \subset P$" can be represented by the *weight of positive evidence* w^+ and the *weight of negative evidence* w^-, which at the current stage can be thought of as simply the number of terms that count as positive and negative evidence in the hypothetical body of binary "ideal experience" defined above. Let us further define the *weight of all evidence* w as the sum of w^+ and w^-, and calculate the weights from the cardinalities of the sets involved (as mentioned previously, the extension and intension of a term are sets of terms).

Definition 8

$$w^+ = |E_S \cap E_P| + |I_P \cap I_S|,$$

$$w^- = |E_S - E_P| + |I_P - I_S|,$$

$$w = w^+ + w^- = |E_S| + |I_P|.$$

Obviously, any two of these three weights suffice to determine the truth value of a statement, and what is measured here is not the extent to which the statement

[3] One should not infer that I am simply sweeping Hempel's paradox under the rug by sacrificing the system's representational and inferential capacities. In NARS 4, the next step of my research, compound terms like $(bird - raven)$, which corresponds to "a bird other than a raven", will be introduced into the system. I will argue, in my future writings about NARS 4, that "absolute negative terms", such as "non-raven", are unsuitable for intelligent reasoning systems like NARS.

matches an objective “state of affairs”, but the extent to which the statement is supported/refuted by available evidence. Such a truth value is subjective in the sense that it is “from the system’s point of view”, but objective in the sense that it is precisely determined by given evidence (Keynes, 1921).

An important feature of the above definition of truth value is that the “extensional factor” and the “intensional factor” are merged.

It is possible to develop extensional or intensional logics separately (Wang, 1994c), and it is easy to see that a pure extensional logic and a pure intensional logic must have different inference rules. As was stated previously, “$S \subset P$” means, when it is understood *extensionally*, that P inherits S’s instances; but when it is understood *intensionally*, the same relation means that S inherits P’s properties. Therefore, if “$S \subset M$” is completely false and “$M \subset P$” is completely true, what can be derived from them is different in the two logics. In the extensional logic, the premises are understood as “S and M have no common instances, and all instances of M are also instances of P”. From these two relations, we cannot decide whether S and P have common instances. On the other hand, in the intensional logic, the premises are understood as “S and M have no common properties, and all properties of P are also properties of M”, which implies that “S and P have no common properties”. Symmetrically, if “$S \subset M$” is completely true and “$M \subset P$” is completely false, the extensional implication is “S and P have no common instances”, and there is no intensional implication.

Though the extensional logic and the intensional logic, defined in this way, are different, formally they are isomorphic to each other, much as union and intersection are duals of each other in set theory. This isomorphism is described in (Wang, 1994c), and it comes directly from the “dual” definitions of extension and intension in NARS (Definition 5).

The dual definitions of extension and intension make it possible for NARS to treat them *uniformly*, as, for instance, in the above definition of "weight of evidence". We need systems to deal with them together, because the coordination of the extensions and intensions of concepts is an important principle in the development of human cognition (Inhelder and Piaget, 1969), and when evidence is used to judge a conceptual relation, whether the evidence is extensional or intensional is often irrelevant or unimportant. We often determine the extension (set of instances) of a concept according to its intension (set of properties), or the other way around, and seldom judge a relation between concepts by considering the extensional or intensional factor *only*, especially when the system has insufficient knowledge and resources. In the next chapter, we will discuss how such a uniform treatment is carried out by the inference rules of NARS.

Though in principle all the information that we want to put into a truth value is representable in the $\{w^+, w\}$ pair, it is not always natural or convenient for the purposes of NARS. Instead of using absolute measurements, we often prefer relative measurements, such as real numbers in the interval [0, 1]. Fortunately, it is easy to define relative measurements in terms of these weights of evidence.

Definition 9 *The* frequency *of a judgment, f, is w^+/w.*

Because w can be thought of as the number of times that the proposed inheritance relation is checked, with w^+ being the number of times that the relation is confirmed, f indicates the "success rate" of the inheritances (of extension and intension) linking the two terms, according to the experience of the system. Obviously, this measurement is closely related to probability and statistics, and often arises in everyday life. However, it is still different from probability under traditional interpretations (logical, frequentist, and subjective, as defined in (Kyburg, 1974)) because it is determined by finite empirical evidence.

Another fundamental difference between probability and frequency is: probability is traditionally interpreted as being solely about extensions of sets. For example, if we say "the probability of '$S \subset P$' is p", p is usually understood as, or closely related to, $|S \cap P|/|S|$, where S and P denote sets of objects. However, as was described earlier, frequency (in NARS) involves both extensional and intensional aspects of the two terms. Therefore, it can be used to represent intensional quantities, such as fuzziness, typicality, and so on. (Wang, 1996b) is a detailed discussion of how to interpret fuzziness and represent it in NARS, and of how this approach is different from fuzzy logic (Zadeh, 1965).

As examples of the above difference, when the system assigns a relatively low frequency value to "$penguin \subset bird$", it does not mean that (extensionally) only a small percentage of penguins are birds, but that (intensionally) the concept "penguin" lacks some properties that the concept "bird" has. According to set theory, the probability of "$cat \subset dog$" should be 0, because no "cat" is also a "dog". Things are otherwise, however, in NARS, because the system may give "$cat \subset dog$" a positive frequency to indicate that, as far as the system knows, "cat" does share properties with "dog". When it is translated into English, such a judgment becomes "Cat has some dogness", rather than "Some cats are dogs".

Though we sometimes explicitly distinguish extensional and intensional relations, as above, we often blur the two together. Indeed, psychological research shows that people use (intensional) "representativeness" as (extensional) "probability", which is usually classified as an error caused by the use of heuristics (Tversky and Kahneman, 1974). However, since NARS assumes the insufficiency of knowledge and resources, and attempts to coordinate the extension and intension of each term, it uses a truth value both extensionally and intensionally. As a result, some "fallacies" (according to extensional theories) become valid modes of reasoning in NARS (Wang, 1996a).

Examples of this phenomenon will be discussed in Chapter 6.

To represent a truth value by a frequency value is not enough for NARS: in addition, we (as well as a computer system) need to know the value of w in order to figure out how to revise frequency in response to new evidence (Wang, 1993a). Can we find a natural way to represent the necessary information in the form of a relative measurement, or, more specifically, as a *ratio*? In the next chapter, we will see why we do not want to use w directly (though it is possible), but prefer a measurement in the $[0, 1]$ interval.

One attractive idea would be to define a “second-order probability”. The frequency defined above can be considered to be an estimate of the “first-order probability” (of the given inheritance relation), and the second-order probability is used to describe how good the first-order estimate is. In fact, several projects have been based on this approach (Fung and Chong, 1986; Gaifman, 1986; Paaß, 1991). However, there are problems concerning how to interpret the second value, and it is unclear how useful it is (Kyburg, 1988; Pearl, 1988). For our current purposes, under the assumption of insufficient knowledge, it makes little sense to talk about the “probability” that “the frequency is an accurate estimate of an ‘objective first-order probability’ of the inheritance relation” (Wang, 1994a). Because NARS is always open to new evidence, it is simply impossible to decide whether the frequency of a judgment will converge to a limiting value in the infinite future, not to mention what that value will be.

However, it makes perfect sense to talk about the *near* future. What the system needs to know, from the value of w, is how *sensitive* a frequency will be to new evidence; then the system can use this information to make a choice among competing judgments. If we limit our attention to a future of fixed horizon, we can represent the information in w in a *ratio* form (Wang, 1994a).

Let us introduce a positive constant k, whose value can be metaphorically thought of as the distance to the (temporal) horizon, in the sense that k can be thought of as the number of times we will still test the given inheritance relation. With this new notion of "horizon", measured by k, we can define a new measurement — confidence, in terms of the weight of all evidence w.

Definition 10 *The* confidence *of a judgment, c, is $w/(w+k)$.*

Intuitively, confidence is the ratio of the weight of all current evidence to the weight of all evidence in the near future. It indicates how much the system knows about the particular inheritance relation, and thus is similar to Shafer's "reliability" (Shafer, 1990) or Yager's "credibility" (Yager, 1991). Since k is a constant, the more the system currently knows about the inheritance relation (i.e., the bigger w is), the more confident the system is about the frequency, since any effect of evidence arriving in the near future will be relatively smaller (we will see how c actually works in the revision operation in the next chapter). For our current purposes, k can be any positive number.

Though c is in $[0, 1]$, and can be considered to be a ratio, and is at a "higher level" than f (in the sense that it indicates the stability of f), it cannot be interpreted as a second-order probability in the sense that it is the probability of the judgment "the (real or objective) probability of the inheritance relation is f" (Wang, 1994a). The higher the confidence is, the harder it will be for the frequency to be changed by new evidence, but this does not mean that the judgment is "truer", or "more accurate", as some psychologists mean when they use the term "confidence" (Einhorn and Hogarth, 1978), because in an open system like NARS, the concept of a real or objective probability does not exist.

It is easy to calculate w and w^+ from f and c, as well as vice versa, and therefore the truth value of a judgment can also be represented as a pair of ratios $< f, c >$

(Wang, 1993a).

Interestingly, there is a third way to represent a truth value in NARS: as an interval (Wang, 1994b). Let us first define two measurements.

Definition 11 *The* lower frequency *of a judgment, l, is $w^+/(w+k)$; the* upper frequency *of a judgment, u, is $(w^+ + k)/(w+k)$.*

Here k is the same constant as was introduced above. Obviously, no matter what happens in the *near future*, the "success frequency" will lie in the interval $[l, u]$ after the constant period. This is because the current frequency is w^+/w, so in the "best" case, when all evidence in the near future is positive, the new frequency will be $(w^+ + k)/(w+k)$; in the "worst" case, when all evidence in the near future is negative, the new frequency will be $w^+/(w+k)$.

This measurement has certain intuitive aspects in common with other interval-based approaches (Bonissone, 1987; Kyburg, 1988; Weichselberger and Pöhlmann, 1990). For example, the *ignorance* about where the frequency will be can be represented by the *width* of the interval (in NARS it happens to be $1 - c$, so *ignorance*, i, and *confidence*, c, are complementary to each other). However, in NARS the interval is defined as the range in which the frequency will lie in the *near* future, rather than in the *remote* future. In this way, some theoretical problems can be avoided. Just as in the earlier discussion of "second-order probability", it is impossible for an open system (as defined previously) to determine the lower or upper bound of an "objective probability".

Now we have three functionally equivalent ways to represent a truth value (or, equivalently, the uncertainty)[4] about an inheritance relation:

[4] According to model-theoretic semantics, or from an observer's point of view, a measurement

1. as a pair of *weights* $\{w^+, w\}$, where $w \geq w^+ \geq 0$;[5] or

2. as a pair of *ratios* $<f, c>$, where both f and c are in $[0, 1]$; or

3. as an *interval* $[l, u]$, where $0 \leq l \leq u \leq 1$.

Three types of brackets ("{}", "<>", and "[ ]") are used in this dissertation for the three forms of truth value, respectively.

Because NARS is designed under the assumption of insufficient knowledge and resources, all judgments within the system are supported by finite evidence — that is, w is positive and finite. For truth values represented in the other two forms, this requirement translates into $0 < c < 1$ and $l < u$, $u - l < 1$.

Beyond the normal truth values, there are two limiting cases useful for the interpretation of truth values and the definition of inference rules:

Null evidence: This is represented by $w = 0$, or $c = 0$, or $u - l = 1$, and of course means that the system knows nothing at all about the inheritance relation.

Full evidence: This is represented by $w = \infty$, or $c = 1$, or $l = u$. It means that the system already knows everything about the statement — no future modification of the truth value is possible.

Formulas for interconversion among the three truth-value forms are displayed in Table 3.1.

of uncertainty (or a degree of belief) is different from a truth value — you can strongly believe a false statement. However, according to experience-grounded semantics, or *within* a system that has insufficient knowledge, both truthfulness and certainty are judged according to available evidence, and the difference between the two notions is not important — I cannot strongly believe a false statement.

[5] Weights of evidence can be extended from integers to rational numbers by allowing evidence with fractional weights. The extension to real values is straightforward, using continuity arguments.

	$\{w^+,\ w\}$	$<f,\ c>$	$[\,l,\ u\,]$
$\{w^+,\ w\}$		$w^+ = k\frac{fc}{1-c}$ $w = k\frac{c}{1-c}$	$w^+ = k\frac{l}{u-l}$ $w = k\frac{1-(u-l)}{u-l}$
$< f,\ c >$	$f = \frac{w^+}{w}$ $c = \frac{w}{w+k}$		$f = \frac{l}{1-(u-l)}$ $c = 1-(u-l)$
$[l,\ u]$	$l = \frac{w^+}{w+k}$ $u = \frac{w^+ + k}{w+k}$	$l = fc$ $u = 1-c(1-f)$	

Table 3.1: Relations among uncertainty measurements.

This table can be easily extended to include w^- (the weight of negative evidence) and i (degree of ignorance). In fact, any valid (not inconsistent or redundant) assignments to any two of the eight measurements (for example, setting $w^+ = 3.5$ and $i = 0.1$, or setting $f = 0.4$ and $l = 0.3$) will uniquely determine the values of all the others. Therefore, the three forms of truth value can even be used in a mixed manner.

Having several closely related forms and interpretations for truth values (uncertainties) has the following advantages (Wang, 1995c):

1. It gives us a better understanding of what a truth value really means in NARS, since we can explain it in different ways. The mappings also give us interesting relations among the various uncertainty measurements.

2. It provides a user-friendly interface. If the environment of the system consists of human users, the uncertainty of a statement can be expressed in different ways, such as, "I've tested it w times, and in w^+ of them it was true", or "Its past success frequency was f, and the confidence was c", or "I'm sure that its success frequency will remain in the interval $[l,\ u]$ in the near future". We can

maintain a single form as the internal representation, and, using the mappings in the above table, translate it into/from the other two for interface purposes.

3. It makes the designing of inference rules easier. For each rule, there should be a function that calculates the truth value of the conclusion from the truth values of the premises, with different rules of course equipped with different functions. As we will see in the next chapter, for some rules it is easier to choose a function if we treat the truth values as *weights*, while for other rules we may prefer to treat them as *ratios* or *intervals*. Clearly, though, no matter which form and interpretation are used, the information carried is precisely the same.

4. It facilitates the comparison between measurements in NARS and the uncertainty measurements of various other approaches, because different forms capture different intuitions about uncertainty. See (Wang, 1993a; Wang, 1994a; Wang, 1994b; Wang, 1995b; Wang, 1996b) for examples.

Because "$S \subset P <1, 1>$" is reflexive and transitive, it is identical to "$S \sqsubset P$" — that is, "$\sqsubset$" is the limit of "$\subset$" as both w and w^+ go to infinity, while w^- remains bounded. In this special case, the effects of negative evidence and future evidence can be ignored. Therefore, "$\sqsubset$" plays a double role in explaining the meaning of "$\subset$": on the one hand, a "$\subset$" relation can be seen as a summary of a set of "$\sqsubset$" relations; on the other hand, a "$\sqsubset$" relation is also the limit of a "$\subset$" relation, so "$\subset$" is a "partial, or imperfect, $\sqsubset$". Understanding this double role is crucial in the design of the inference rules (to be introduced in the next chapter).

In this section we have defined the truth values of sentences in **L** by the system's experience, which is still represented in **L0** (the binary language defined in the previous section), now known as a subset of **L**. Similarly, we can extend the concept of "meaning". For a system whose knowledge is represented in **L**, the meaning of a term still consists of the term's extensional and intensional relations with other terms, as in **L0**. The only difference is that the definition of extension and intension is modified as follows:

Definition 12 *A judgment "$S \subset P < f, c >$" states that S is in P's extension and that P is in S's intension, with the truth value of the judgment specifying their degrees of membership.*

According to this definition, extensions and intensions in NARS are no longer ordinary sets with well-defined boundaries. They are similar to fuzzy sets (Zadeh, 1965), because terms belong to them to different degrees. What makes them different from fuzzy sets is how the "membership" is measured (in NARS, two numbers are used) and interpreted (in NARS, it is experience-grounded) (Wang, 1996b).

Now we have finished our basic task in semantics. Given any set of sentences of **L0** as the experience of a system, we can determine the truth values of sentences and the meanings of terms in **L**. Because the meaning of the "$\sqsubset$" relation in **L0** is completely determined by its two properties, reflexivity and transitivity, the relation is used as a semantic primitive to define the truth values of sentences and the meanings of terms in **L**. Since **L0** is a small subset of **L**, we say that this is a "bootstrapping" way to establish an experience-grounded semantics for **L** as a whole.

3.4 Experience: ideal vs. real

According to the assumption of insufficient knowledge, in NARS the confidence of a sentence cannot reach 1 (which would mean the system had infinite evidence about the sentence), but can approach it as a limit. Therefore, sentences like "$S \sqsubset P$" cannot really appear in the system's experience. However, this does not prevent us from using **L0** to construct an "ideal experience" for semantic purposes. For example, if there is a sentence within the system's knowledge base with the form "$S \subset P$ $<0.75,\ 0.80>$", then from the relationship between truth value and weight of evidence (and assuming $k = 1$), we get $w = 4$, $w^+ = 3$. Therefore, the system believes the relation "$S \subset P$" *as if* it had tested the relation four times (by checking common elements of the extensions or the intensions of the two terms[6]), in which the relation had been confirmed three times, and failed once. This does not imply, of course, that the system actually got the truth value by carrying out such tests — such absolute certainty can never be be obtained in real life. Indeed, the system may have checked the relation more than four times in less-than-ideal situations (i.e., with results represented by judgments whose confidence values are less than 1), or the conclusion may have been derived from other knowledge, or even directly provided by the environment. But no matter how the truth value $<0.75,\ 0.80>$ is generated in practice (there are infinitely many ways it could arise), it can always be *understood* in a unique way, as stated above.

In NARS, the concept of ideal experience is used to *define* truth values and to derive inference rules, while actual experience is used to numerically *calculate* truth values. The distinction between these two types of experience is useful, because "numerical statements are meaningful insofar as they can be translated, using the mapping conventions, into statements about the original qualitative structure" (Krantz, 1991). In other words, "ideal experience" is being used here as an "ideal meter-stick" to measure degrees of truth. Like all measurements, though its unit is defined in an idealized situation, it is not used only in idealized situations.

Another factor that makes actual experience differ from ideal experience is the insufficiency of resources. Due to the lack of memory, some of the system's experience

[6] According to my experience-grounded semantics, such checks are not probings in the physical world, but inspections of available knowledge.

will be forgotten; due to the lack of time, some of the system's experience will be ignored. Consequently, the truth value of a sentence or the explicit meaning of a term (i.e., its revealed relations with other terms) is usually based on *partial experience*, or a *section* of the system's experience (which is a stream of input sentences in **L**).

As was explained above, this fact makes the real situation much more complex than the ideal situation, but it does not prevent us from saying that the truth value of a sentence summarizes its evidential support, and that the meaning of a term derives from its experienced relations with other terms. As was stated before, the function of semantics is to help us to design the system and to understand its language. Now we can see that both of these goals can be achieved by defining truth value and meaning in terms of the system's ideal experience. It needs to be stressed, however, that semantics is *not* about how concrete truth values and meanings are calculated by the system — that function is carried out by the inference rules, designed according to the semantics. We will introduce these rules in the next chapter.

One important character of experience-grounded semantics is its dynamic and subjective nature (Wang, 1995a). Obviously, the truth value of a sentence changes dynamically in NARS, due to the arrival of new experiences. The system's inference activity also changes the truth values of sentences by combining evidence from different sections of the experience. Since truth values are based on the system's experience, they are intrinsically *subjective.* To be more precise, the system's knowledge is not an objective description of the world, but a summary of its own experience, so it is from the *system's point of view.* It is to be expected that two systems in precisely the same environment would have different knowledge, obtained from their different individual experiences.

To say that truth values are dynamic and subjective does not mean that they are arbitrary. As Quine said, "Observations are the boundary conditions of a system of beliefs." (Quine and Ullian, 1970) Different systems in the same environment can achieve a certain degree of "objectivity" by communicating with one another and thus sharing experience. However, here "objective" means "common" or "unbiased", rather than "observer-independent" — the common knowledge is still limited by the experiences of the systems involved.

3.5 Grammar

The basic inheritance relation "$\subset$" is not the only meaningful inheritance relation. In NARS, a *symmetric* inheritance relation, "$=$", and a *singular* inheritance relation, "$\in$", are derived from "$\subset$". Intuitively, they correspond to the *similarity* relation and the *membership* relation between two concepts, respectively. Like "$\subset$", these two relations are also interpreted both extensionally and intensionally. For example, "$creature = animal$" means that the terms "creature" and "animal" have the same instances and the same properties; "$Tweety \in bird$" means both that "Tweety" is an individual instance of "bird" and that "Tweety" has "birdness" — that is, it has the properties of a typical bird. "$S = P$" is a summary of "$S \subset P$" and "$P \subset S$". By this definition, evidence (positive and negative) for either "$S \subset P$" or "$P \subset S$" is also evidence for "$S = P$". Obviously, this relation is symmetric — that is, "$S = P$" is equivalent to "$P = S$". The *singular* inheritance relation is used when we prefer to treat the subject term as an individual or proper noun. For example, "A dove is a bird" is represented as "$dove \subset bird$", but "Tweety is a bird" is represented as "$Tweety \in bird$". By considering the set that contains a unique member $Tweety$, written as "$\{Tweety\}$", it is easy to see that "$Tweety \in bird$" can be rewritten as "$\{Tweety\} \subset bird$".

A sentence in **L**, such as "$S \subset P$", is used in NARS to represent a *question*.

To answer such a question means to derive a judgment about the given inheritance relation, such as "$S \subset P < f, c >$", from available knowledge, under the current resource constraints. Another type of question is characterized by the presence of a question mark "?" in the position of a term — for example, "$? \in P$". To answer such a question means to find a term that has the required inheritance relation with the given term, such as "$S \in P <f, c>$". Here we have not mentioned what counts as a *good* answer to a question. Obviously, "$tiger \in bird < 0, 0.99 >$" ("A tiger is not a bird.") is a pretty bad answer to the question "$? \in bird$" ("What is a bird?"), although it is a valid answer according to the above definition. We will discuss this issue in the next chapter.

In this chapter, a formal language **L** has been defined. In the next chapter, some inference rules that work with the language will be introduced. Together, the two chapters define a non-axiomatic logic. I call the logic NAL2, since it is an extension of NAL1, presented in (Wang, 1994c). Table 3.2 gives the grammar of NAL2.

$$\begin{aligned}
<\!\mathit{judgment}\!> \;&::=\; <\!\mathit{term}\!><\!\mathit{be}\!><\!\mathit{term}\!><\!\mathit{truth\text{-}value}\!> \\
<\!\mathit{question}\!> \;&::=\; <\!\mathit{term}\!><\!\mathit{be}\!><\!\mathit{term}\!> \mid\; ?<\!\mathit{be}\!><\!\mathit{term}\!> \mid <\!\mathit{term}\!><\!\mathit{be}\!>? \\
<\!\mathit{be}\!> \;&::=\; \subset \mid = \mid \in \\
<\!\mathit{term}\!> \;&::=\; <\!\mathit{word}\!> \\
<\!\mathit{word}\!> \;&::=\; <\!\mathit{letter}\!> \mid <\!\mathit{letter}\!><\!\mathit{word}\!> \mid <\!\mathit{word}\!>\text{-}<\!\mathit{word}\!> \\
<\!\mathit{letter}\!> \;&::=\; A|B|\cdots|Y|Z|a|b|\cdots|y|z
\end{aligned}$$

Table 3.2: Grammar of NAL2.

As was described previously, there are (at least) three ways to represent the truth value of a judgment: by weight of evidence, by frequency and confidence, and by frequency interval.

4

Inference Rules

In this chapter, the inference rules used in NAL2 are introduced. Each rule takes as input two premises, and from them derives a conclusion. When the two premises are both judgments, the conclusion is also a judgment, whose truth value is determined by the truth values of the premises.

4.1 Revision and choice

As was stated above, it is possible (in fact, it is usually the case) for the judgments in the memory of NARS to conflict with each other, in the sense that at a given time, there are two coexistent judgments that attach different truth values to the same inheritance relation, in the following manner:

$$S \subset P \, \{w_1^+, \, w_1\}$$

$$S \subset P \, \{w_2^+, \, w_2\}$$

where the truth values are represented as weights of evidence.[1]

Such conflicts arise from the fact that the judgments involved are based on different sections of the experience of the system, say K_1 and K_2, both of which correspond to sets of input judgments. The concept "section of experience" is defined recursively, as follows:

Definition 13 *If j is an input judgment that appears in the system's experience, with a unique serial number N, it is based on the section of experience $\{N\}$.*

Definition 14 *If j is not an input judgment but is derived according to an inference rule of NAL2 from premises j_1 and j_2, which are based on the sections of experience K_1 and K_2 respectively, then j is based on section $K_1 \cup K_2$.*

According to the semantics of NARS, as long as K_1 and K_2 have no common elements, the two bodies of evidence supporting the two conflicting judgments do not overlap with each other (that is, no piece of evidence is a contributor to both of the two premises). In this situation, a *revision* rule is applied to the two premises, and the conclusion derived should be

$$S \subset P\,\{w_1^+ + w_2^+,\ w_1 + w_2\},$$

where the evidence from different sections of experience is summarized, or pooled (Wang, 1994b). The use of addition here is justified by the semantics introduced in the previous chapter. Since K_1 and K_2 have no common elements, the ideal experiences measured by $\{w_1^+,\, w_1\}$ and $\{w_2^+,\, w_2\}$ can be combined by adding the

[1] In contrast to first-order predicate logic, where any conclusion whatsoever can be derived from a pair of propositions that differ only in their truth values, in term logics a conflict is a local problem, in the sense that not all results are affected. See the following discussion of the various types of inference rules in NAL2.

weights of (positive, negative, and all) evidence, respectively. In general, the revision rule is used to accumulate evidence, even if the two pieces of evidence support identical judgments.

Given the relations among different forms of truth value in Table 3.1, it is easy to convert the above-given truth-value function for the revision rule into a formula in terms of frequency and confidence:

$$\begin{aligned} F_{rev}: \quad f &= [c_1(1-c_2)f_1 + c_2(1-c_1)f_2] \;/\; [c_1(1-c_2) + c_2(1-c_1)] \\ c &= [c_1(1-c_2) + c_2(1-c_1)] \;/\; [c_1(1-c_2) + c_2(1-c_1) + (1-c_1)(1-c_2)] \end{aligned}$$

So far, so good. However, things are more complex than this, because with insufficient resources, NARS cannot maintain a complete record of the supporting experience for each judgment; after all, the evidence in support of a given judgment could involve arbitrarily large regions of memory and arbitrarily long chains of operations.

Therefore the "overlapping-evidence recognition problem" cannot be completely solved by a system with insufficient resources. Obviously, this limitation holds also for human beings: we could not possibly remember all evidence that supports each judgment we make. Nevertheless, NARS needs to be able to handle this problem somehow; otherwise, as Pearl points out, "a cycle would be created where any slight evidence in favor of A would be amplified via B and fed back to A, quickly turning into a stronger conformation (of A and B), with no apparent factual justification." (Pearl, 1988)

The NARS strategy for dealing with this fundamental problem is to record only a *constant-sized fragment* of the experience supporting each judgment, and to use such fragments to determine heuristically whether two judgments are based on overlapping evidence. As was mentioned above, each input judgment is automatically assigned a unique serial number when accepted by the system. In each inference step, the conclusion is assigned a list of serial numbers constructed by interleaving its parents' (the premises') serial-number lists, and then truncating that list at a certain length. For example, suppose the maximum length for serial-number lists is 4. In this case, two judgments whose serial-number lists overlap will have a parent or grandparent judgment in common. Now the revision rule is applied only if the two premises' serial-number lists have no common elements, meaning that they are related, if at all, more than two generations ago. This mechanism is only an approximation, of course. Though not perfect, it is a reasonable solution when resources are insufficient, and "reasonable solutions" are exactly what we expect from a non-axiomatic system.[2] It is also similar to the strategy of the human mind, since we usually have impressions about where our more recent judgments come from, but such impressions fade quickly and are far from complete and accurate.

What should NARS do when two conflicting judgments are based on overlapping evidence? Ideally, we would like to record the precise contribution of each input judgment, and then to subtract the weight of the overlapping section from the truth value of the conclusion, so that nothing is double-counted. Unfortunately, this is impossible, because the experience recorded for each judgment is incomplete, as has just been explained. Nevertheless, NARS needs to be able to handle this situation. For example, the two conflicting judgments may be candidate answers to a question (recall the discussion at the end of the previous chapter). If it is impossible to combine them, then NARS needs to make a choice between the two. In the current situation, the *choice* rule is very simple: the judgment having a *higher confidence* (no matter what its *frequency* is) should be taken as the better answer, the idea being that if an

[2] This is an example of what it means to develop a model under the assumption of insufficient knowledge and resources. Without such an assumption, we could solve the problem perfectly by completely recording all supporting experience, using a recursive technique, then checking for intersection of the records.

adaptive system must make a choice between conflicting judgments, the one based on more experience has higher priority.

To make a choice between two or more competing answers for a question is not always this simple. Let us say that the system is asked the question "$S \subset ?$", meaning that it should come up with a term T that is a "typical element" in the intension of S (not S itself, of course). Ideally, the best answer would be provided by a judgment "$S \subset T < 1, 1 >$" (i.e., frequency 1 and confidence 1). But of course this is impossible, because confidence can never reach 1 in NARS. Therefore, we have to settle for the best answer the system can find under the constraints of available knowledge and resources.

Suppose the competing answers are

$$S \subset T_1 \ < f_1, \, c_1 >$$

$$S \subset T_2 \ < f_2, \, c_2 > .$$

Which one would be better? Let us consider some special cases first:

1. $c_1 = c_2$, meaning that the two answers are supported by the same amount of evidence. For example, both come from statistical data of 100 samples[3]. Obviously, the answer with the *higher frequency* is preferred, since that inheritance relation has more positive evidence than the other.

2. $f_1 = f_2 = 1$, meaning that all available evidence is positive. Now the answer with *higher confidence* is preferred, since it is more strongly confirmed by experience.

[3] As was stated in the previous chapter, each sample can be represented by a pair of sentences in **L** — "$M \subset T_1 \ < 1, 0.8 >$" and "$M \subset S \ < 1, 0.8 >$" makes M a piece of positive evidence for "$S \subset T_1 \ < f_1, \, c_1 >$". This issue will be discussed in more detail when the induction and abduction rules are introduced later.

3. $f_1 = f_2 = 0$, meaning that all available evidence is negative. Now the answer with the *lower confidence* is preferred, since it is less strongly refuted by the experience. Of course such an answer is still a bad one because of its negative nature, but it may be the best (the least negative) answer the system can find for the question.

From these special cases, we can see that to set up a *general rule* to make a choice among competing judgments, we need somehow to combine the two numbers in a truth value into a single measurement. The current situation is different from the previous one. "$S \subset T_1 < f_1, c_1 >$" and "$S \subset T_2 < f_2, c_2 >$" do not conflict with each other — they are about different relations — but they compete for being the "best supported intensional relation of S".

In NARS, *expectation*, e, is defined for this purpose. Different from a truth value (which is used to record past experience), an expectation (of a judgment) is used to predict future experience. "$e = 1$" means the system is absolutely sure that the inheritance relation under consideration will always be confirmed by future experience; "$e = 0$" means it will always be refuted; and "$e = 0.5$" means the system considers it equally likely to encounter a positive or a negative instance. Intuitively, e is similar to *subjective probability* (Kyburg, 1974); it can be interpreted as the system's estimate of the future "inheritance frequency", or equivalently, as a bet the system will accept about a future "inheritance test". Under the assumption of insufficient knowledge, in NARS e takes values in the open interval (0, 1), with 0 and 1 as unattainable limits. In the case of the competing answers described above, the system takes the one whose expectation is higher.

To calculate e from $< f, c >$, we can see that under the assumption that the system makes extrapolations from its (past) experience, it would be natural to use f as e's "first-order approximation". However, such a *maximum-likelihood estimate* is not good enough when c is small (Good, 1965). For example, if a hypothesis has been tested only once, it would not make sense to set one's expectation to 1 (if the test was a success) or to 0 (if the test was a failure).

Intuitively, e should be more "conservative" (i.e., closer to 0.5, the "no-preference point") than f, to reflect the fact that the future may be different from the past. Here is where the confidence c affects e — the more evidence the system has accumulated, the more confident the system is (indicated by a larger c) that its predicted frequency e should be close to its experienced frequency f. Therefore, it is natural to define

$$e = c(f - 0.5) + 0.5.$$

In particular, when $c = 1$ (full evidence), $e = f$; when $c = 0$ (null evidence), $e = 0.5$. Alternatively, this equation can be rewritten as $c = (e - 0.5)/(f - 0.5)$ (when $f \neq 0.5$), showing that c indicates the ratio of e's and f's distances to 0.5.

To express the definition of e in the other two forms of truth value leads to interesting results.

When the truth value is represented as an interval, from the interconversion formulas in Table 3.1, we get

$$e = 0.5(l + u)$$

Thus e is precisely the *expectation of the future frequency* — that is, the midpoint of the interval in which the frequency will lie, in the near future.

When the truth value is represented as weights of evidence, from the mappings we get

$$e = (w^+ + k/2)/(w + k)$$

which is a continuum (i.e., a family) of functions with k as a parameter. This formula turns out to be closely related to what has been called the "beta-form based continuum" (with positive and negative evidence weighted equally) (Good, 1965), and the "λ-continuum" (with the "logical factor", or prior probability, being 1/2) (Carnap, 1952). Though interpreted differently, the three continua share the same formula and make identical predictions. All three continua have *Laplace's law of succession* as a special case (when $k = 2$), where the probability of success on the next trial is estimated by the formula $(w^+ + 1)/(w + 2)$.

Now we can see how the choice of the constant k can influence the behavior of a system (Wang, 1995b). Let us compare a system A_1 with $k = 1$ and a system A_2 with $k = 10$. The problem is to make a choice between two competing answers "$S \subset P_1 \ \{w_1^+, w_1\}$" and "$S \subset P_2 \ \{w_2^+, w_2\}$" (where the truth values are represented as weights of evidence). It is easy to see that when $w_1 = w_2$ or $w_1^+/w_1 = w_2^+/w_2$, the two systems make the same choice. It is only when a system needs to make a choice between a higher f and a higher c that the value of k will matter. For example, let us suppose that $w_1^+ = w_1 = 2, w_2^+ = 5$, and $w_2 = 6$. In this situation, in A_1, $e_1 = (2 + 0.5)/(2 + 1) \approx 0.83$, $e_2 = (5 + 0.5)/(6 + 1) \approx 0.79$, and thus A_1 will choose the first answer (since all of its evidence is positive); in A_2, $e_1 = (2+5)/(2+10) \approx 0.58$, $e_2 = (5+5)/(6+10) \approx 0.63$, and thus A_2 will choose the second answer (since it is more fully tested, and its frequency is not much lower than that of the other alternative).

Therefore, k is one of the "personality parameters" of a reasoning system, in the sense that it indicates a certain systematic preference or bias, for which there is no "optimal value" in general. The larger k is, the more "conservative" the system is, in the sense that the system always accepts smaller bets, and makes smaller adjustments when e is reevaluated according to new evidence, than a system having a smaller value of k. This parameter was called the "flattening constant" by Good ((Good, 1965), where he also tried to estimate its value according to certain factors that are beyond our current consideration), and was interpreted by him as a way to choose a prior probability distribution. The same parameter was interpreted by Carnap as the "relative weight" of the "logical factor" (Carnap, 1952).

4.2 Syllogisms

In term logics, when two judgments share a common term, they can be used as premises in an inference rule that derives an inheritance relation between the other two (unshared) terms. Altogether, there are four possible combinations of premises and conclusions, corresponding to the four figures of Aristotle's syllogisms (Aristotle, 1989), three of which are also discussed by Peirce (Peirce, 1931):

1. From "$M \subset P < f_1, c_1 >$" and "$S \subset M < f_2, c_2 >$" to get "$S \subset P < f, c >$". This is Aristotle's *first figure*, and what Peirce called *deduction*. Let us refer to the function that calculates f and c from f_1, c_1, f_2, and c_2 as F_{ded}.

2. From "$P \subset M < f_1, c_1 >$" and "$S \subset M < f_2, c_2 >$" to get "$S \subset P < f, c >$". This is Aristotle's *second figure*, and what Peirce called *abduction* (or *hypothesis*). Let us refer to the function that calculates f and c from f_1, c_1, f_2, and c_2 as F_{abd}.

3. From "$M \subset P < f_1, c_1 >$" and "$M \subset S < f_2, c_2 >$" to get "$S \subset P < f, c >$". This is Aristotle's *third figure*, and what Peirce called *induction*. Let us refer to the function that calculates f and c from f_1, c_1, f_2, and c_2 as F_{ind}.

4. From "$M \subset P < f_1, c_1 >$" and "$S \subset M < f_2, c_2 >$" to get "$P \subset S < f, c >$". This rule, not discussed by Aristotle or Peirce, was called the *fourth figure* by Aristotle's successors (Bocheński, 1970). I will call it *exemplification*. Let us refer to the function that calculates f and c from f_1, c_1, f_2, and c_2 as F_{exe}.

Because of the conceptual extension of the notion of "truth value", and because terms are interpreted both extensionally and intensionally, the syllogisms in NARS are quite different from Aristotle's and Peirce's, though still related to them.

How to determine mathematical formulas for the four types of inference just described? Basically, their general properties are dictated by the semantics of NARS. In this section, we will see that our definition of the notion of "truth value" sets precise boundary conditions on the functions. Because many of the quantities involved are real numbers in $[0, 1]$, we can infer the form of the desired functions solely from the boundary conditions by making use of the so-called *Triangular norm* (T-norm) and *Triangular conorm* (T-conorm) (Bonissone and Decker, 1986; Dubois and Prade, 1982; Schweizer and Sklar, 1983).

T-norm, $T(x_1, x_2)$, and T-conorm, $S(x_1, x_2)$, are distinct binary operations defined on real numbers in [0, 1]. Each of them is both commutative and associative, and monotonic in each variable. T-norm has boundary conditions satisfying the truth tables of the logical operator AND, and T-conorm those of OR. Because each is commutative and associative, each of them can be extended to take an arbitrary number of arguments in the following way, precisely analogous to how addition and multiplication are extended from two arguments to an arbitrary number of arguments:

$$T(x_1, \ldots, x_n) = T(T(x_1, \ldots, x_{n-1}), x_n),$$

$$S(x_1, \ldots, x_n) = S(S(x_1, \ldots, x_{n-1}), x_n).$$

The usage of T-norm and T-conorm in NARS is different from their usual usage in uncertainty calculus (Bonissone and Decker, 1986; Dubois and Prade, 1982), where they are used to determine the degree of certainty of the *conjunction* and *disjunction* of two propositions, respectively. In NARS, the T-norm function $y = T(x_1, \ldots, x_n)$ is used when a quantity y is *conjunctively* determined by two or more other quantities $x_1, \ldots, x_n$ — that is, $y = 1$ if and only if $x_1 = \cdots = x_n = 1$, and $y = 0$ if and only if $x_1 = 0$ or ... or $x_n = 0$; similarly, the T-conorm function $y = S(x_1, \ldots, x_n)$ is used when a quantity y is *disjunctively* determined by two or more other quantities $x_1, \ldots, x_n$ — that is, $y = 1$ if and only if $x_1 = 1$ or ... or $x_n = 1$, and $y = 0$ if and only if $x_0 = \cdots = x_n = 0$. These quantities are not about the *conjunction* or *disjunction* of two judgments.[4]

Intuitively, a variable y is conjunctively determined by variables $x_1, \ldots, x_n$ when all the x's are its *necessary* conditions, or numerically, if y is never bigger than any of them. Similarly, y is disjunctively determined by $x_1, \ldots, x_n$ when all the x's are its *sufficient* conditions, or numerically, it is never smaller than any of them. T-norm and T-conorm can be applied in situations where a quantity is determined by several factors, where we wish the boundary condition to be satisfied, and where no one factor is more important than any of the others.

There are an infinite number of ways of numerically satisfying the prescribed conditions on T-norm and T-conorm. For the current purpose, we want them to be continuous and strictly increasing, so that any upward (downward) change in any argument will cause an upward (downward) change in the function value. In (Schweizer and Sklar, 1983) it is proven that all functions satisfying the above conditions are isomorphic to (i.e., can be represented as a monotonic transform of) the "probabilistic" operators (which define, respectively, the probabilities of the union and intersection of two mutually independent events with probabilities a and b):

$$T(a, b) = ab; \quad S(a, b) = a + b - ab.$$

[4] In NAL2, the conjunction or disjunction of two judgments is not defined as a judgment.

It is also shown in (Bonissone and Decker, 1986) that only a small finite subset of the infinite set of possible T-norms and T-conorms will produce significantly different results, if we limit our concern to the "finest level of distinction among different quantifications of uncertainty". Among those representative operators in the small subset, the above pair is the only continuous and strict T-norm and T-conorm. The Schweizer–Sklar and Bonissone–Decker results show that the above T-norm and T-conorm have not been chosen arbitrarily for NARS; although in principle there are other pairs satisfying our requirements, they are usually more complex, and are not significantly different from the above pair.

The above choice is also justifiable in another way. Since in NARS the syllogistic inference rules are applied only to premises based on non-overlapping evidence, and since the frequency and confidence of a judgment are determined by different factors, it follows that f_1, c_1, f_2, and c_2 are *mutually independent* of each other, in the sense that given the values of any three of them, the fourth cannot be determined, or even bounded approximately. This type of mutual independence among arguments is assumed by the probabilistic operators, but not by other representative operators, such as the pair used in fuzzy logic (Bonissone and Decker, 1986): $T(a, b) = min(a, b)$; $S(a, b) = max(a, b)$.

We now proceed to determine the truth-value functions for the four syllogistic inference rules in terms of T-norm and T-conorm, guided by the boundary conditions of extreme cases, where we know what we want to happen.

As defined previously, the *deduction* rule in NARS takes "$M \subset P < f_1, c_1 >$" and "$S \subset M < f_2, c_2 >$" as premises, and derives a conclusion "$S \subset P < f, c >$" from them. Therefore, it extends the "rule of transitivity" in the binary term logic IL, introduced in the previous chapter. Since "$S \subset P < f, c >$" becomes "$S \sqsubset P$" if and only if both f and c are 1, and since T-norm is used in NARS to extend the "and" relation into $[0,1]$, we would let $T(f,c) = T(T(f_1,c_1),T(f_2,c_2))$, which means precisely that "$S \sqsubset P$" is deduced from "$M \sqsubset P$" and "$S \sqsubset M$" — both f and c are 1 if and only if f_1, c_1, f_2, and c_2 are all 1.

More generally, if $T(f_1,c_1) = 1$, then $< f, c >=< f_2, c_2 >$, because here the first premise says that P completely inherits M's extensional relations, among which there is the relation to S. Symmetrically, if $T(f_2,c_2) = 1$, then $<f, c>=<f_1, c_1>$, meaning that S completely inherits M's intensional relation to P.

From these special situations, we derive the confidence value of the conclusion in general: $c = T(c_1,c_2,S(f_1,f_2))$, which means that the confidence of the conclusion is determined conjunctively from the values c_1, c_2, and $S(f_1,f_2)$. The first two arguments of the T-norm are easy to understand — the conclusion gets full evidence only when both of the premises have full evidence, and the conclusion gets null evidence when either of the two premises has null evidence, therefore the boundary condition of T-norm is satisfied. $S(f_1,f_2)$ gets involved here because nothing can be deduced from two completely negative premises (even if their confidences are 1) — from "M and P share no extension or intension" and "S and M share no extension or intension", no evidence (positive or negative) is provided for "$S \sqsubset P$". If f_1 or f_2 is 1 (or identically, $S(f_1,f_2) = 1$), and both c_1 and c_2 are 1, we know that one of the premises is in the "$\sqsubset$" form, which leads to a "complete inheritance" by definition. For example, let us suppose that $f_1 = c_1 = 1$, so that P completely inherits M's extension, including its (extensional) relation with S, from which it follows that the conclusion should be "$S \subset P <f_2, c_2>$". The case for $f_2 = c_2 = 1$ is completely symmetric.

Here it needs to be stressed again that in NAL2 the "$\subset$" relation is not identical to the subset relation in set theory, which is purely extensional. If "$M \subset P <1, 1>$" is interpreted as "Set M is completely included in set P", and "$S \subset M < 0, 1 >$" as "Set S and set M have an empty intersection", it is *invalid* to derive from them

"$S \subset P < 0, 1 >$", which means "Set S and set P have an empty intersection". In NARS, the relation is both extensional and intensional. As a result, when an extensional inheritance exists, it is used as in set theory. On the other hand, if there is no available extensional relation, NARS may use intensional relations to get conclusions, which is impossible in set theory.

Putting the above two formulas together, we get the truth-value function of the deduction rule:

$$\begin{array}{rcccl} F_{ded}: \; f & = & T(f_1, f_2)/S(f_1, f_2) & = & f_1 f_2/(f_1 + f_2 - f_1 f_2) \\ c & = & T(c_1, c_2, S(f_1, f_2)) & = & c_1 c_2 (f_1 + f_2 - f_1 f_2) \end{array}$$

For the sake of continuity, we let $f = 0$ when $f_1 = f_2 = 0$.

In NARS, *abduction* is the inference that from a shared element M of the *intensions* of S and P determines the truth value of "$S \subset P$", and *induction* is the inference that from a shared element M of the *extensions* of S and P determines the truth value of "$S \subset P$". From the symmetry between extension and intension, we know that $F_{abd} = F'_{ind}$, and $F_{ind} = F'_{abd}$, where F'_* is the function gotten by exchanging $< f_1, c_1 >$ and $< f_2, c_2 >$ in the function F_*. Therefore, we only need to discuss one of them, say F_{ind}.

Let us suppose that we definitely know "Tomato is a kind of plant" and "Tomato is a kind of vegetable"; can we then infer whether, or to what extent, vegetable is a kind of plant? If we write all three sentences in first-order predicate logic, and interpret "is a kind of" extensionally as the subset relation, then all we know is that the sets "vegetable" and "plant" have a common subset, "tomato", which tells us nothing about the extent to which "vegetable" is included in "plant". But if we resort to experience-grounded semantics, the situation is different. Here the system is concerned with determining the extent to which the term "vegetable" can be used as the term "plant". Therefore, "tomato" becomes a piece of positive evidence for "Vegetable is a kind of plant", and the truth value of the conclusion reflects the degree of support provided by the evidence, rather than measuring how many vegetables are plants in the real world.

In determining the truth value of "$vegetable \subset plant$" from the common instance *tomato* of its two terms, the truth values of the two premises play different roles. The frequency of "$tomato \subset plant$", f_1, estimates the frequency of the conclusion, since the property "being *plant*" of the specific term *tomato* is taken as a property of the general term *vegetable*. On the other hand, f_2, c_2, and c_1 *conjunctively* determine the extent to which *tomato* can be counted as a piece of relevant evidence for the conclusion. If f_2 or c_2 is 0, *tomato* is not in the extension of *vegetable* at all (so it cannot serve as evidence); also, if c_1 is 0, the first premise provides no information about the relation between *tomato* and *plant*, thus the conclusion gets no support in this case as well. Only when $T(f_2, c_2, c_1) = 1$ does *tomato* become a piece of evidence with weight 1 (because now "$tomato \subset vegetable < f_2, c_2 >$" becomes "$tomato \sqsubset vegetable$"). Since "$w$ is conjunctively determined by f_2, c_2, and c_1" is represented by $w = T(f_2, c_1, c_2)$, we have

$$\begin{aligned} F_{ind}: \quad f &= f_1 \\ c &= f_2c_2c_1/(f_2c_2c_1 + k) \end{aligned}$$

Since F_{abd} is the same as F_{ind} with the roles of $<f_1,\ c_1>$ and $<f_2,\ c_2>$ reversed, we also obtain:

$$\begin{aligned} F_{abd}: \quad f &= f_2 \\ c &= f_1c_1c_2/(f_1c_1c_2 + k) \end{aligned}$$

Under these definitions, the well-known difference between abduction or induction and deduction is preserved: deductive conclusions are usually much more confident (with 1 as their upper bound) than abductive or inductive conclusions (which have $1/(1+k)$ as their upper bound). Here we can see another function of the personality parameter k: to indicate the relative confidence of abductive/inductive conclusions. Intuitively speaking, all intelligent systems (human and computer) need to maintain a balance between the strictness of deduction and the tentativeness of induction and abduction. For certain purposes, it is better to let the "balancing point" remain stable (though not necessarily constant) to ensure self-consistency of the system's behavior. However, there is no single "optimal value" for such a parameter, at least for our current discussions. Comparatively speaking, a system with a small k relies more on abduction and induction, while a system with a large k relies more on deduction.

Using F_{abd} or F_{ind}, we can define a *conversion rule.* In term logics, "conversion" is an inference from a single premise to a conclusion by interchanging the subject and predicate terms (Bocheński, 1970). Now we can see conversion as a special case of abduction by taking "$P \subset S < f_0, c_0 >$" and "$S \subset S < 1, 1 >$" (a tautology) as premises, and "$S \subset P < f, c >$" as conclusion. Using F_{abd}, we obtain the truth-value functions for the conversion rule:

$$\begin{aligned} F_{con}: \quad f &= 1 \\ c &= f_0 c_0 / (f_0 c_0 + k) \end{aligned}$$

We could also derive this same result by seeing conversion as a special case of induction with "$P \subset P < 1, 1 >$" and "$P \subset S < f_0, c_0 >$" as premises.

How can we understand the conversion rule directly? From our interpretation of the truth value, we see that any positive evidence for "$S \subset P$" (the terms in $E_S \cap E_P$

and $I_P \cap I_S$) will also be positive evidence for "$P \subset S$", but any negative evidence concerning the former (the terms in $E_S - E_P$ and $I_P - I_S$) will be irrelevant to the latter (because those terms are not in E_P or I_S). This means that the conclusion can only be confirmed, but never refuted, by conversion. Consequently, $f = 1$ in all situations. As the *weight* of the conclusion, we know that it is at most 1, and this will happen only when $f_0 = c_0 = 1$. In that case, P is in the extensions of both S and P. Therefore, we take $w = T(f_0, c_0)$.

This analysis leads us to the truth-value functions for *exemplification*, the "fourth figure" in (Wang, 1994c). This rule takes the same premises as the deduction rule, but in its conclusion the most general term in the premises, P, becomes the subject, while the most specific term in the premises, S, becomes the predicate. For instance, from "Tomato is a kind of vegetable" and "Vegetable is a kind of plant", by deduction we get "Tomato is a kind of plant", but by exemplification we get "Plant is a kind of tomato". That is why the name "exemplification" is used for this rule — it states that, to a certain extent, "plant" inherits the properties of "tomato", and "tomato" inherits the instances of "plant". As in the case of the conversion rule, no negative evidence for the conclusion can be collected in this way, and the w of the conclusion is determined conjunctively by f_1, c_1, f_2, and c_2. Only when $f_1c_1f_2c_2 = 1$ can the conclusion get support with strength $w = 1$, since then we have "$P \sqsubset S$" — that is, P is in the extensions of both S and P. Therefore, here we have

$$\begin{aligned} F_{exe}: \quad f &= 1 \\ c &= f_1c_1f_2c_2/(f_1c_1f_2c_2 + k) \end{aligned}$$

There is another interesting result. We know that from "$M \subset P < f_1,\ c_1 >$" and "$M \subset S < f_2,\ c_2 >$", NARS can directly get "$S \subset P < f_1,\ f_2c_2c_1/(f_2c_2c_1 + k) >$" by induction. There is also an indirect way to derive "$S \subset P$" from the same premises: via conversion, the second premise yields "$S \subset M < 1, f_2c_2/(f_2c_2 + k) >$"; then, deductively combining this judgment with the first premise, NARS arrives at the conclusion "$S \subset P < f_1,\ f_2c_2c_1/(f_2c_2 + k) >$". Compared with the direct result, this indirect conclusion has the same frequency value, but a lower confidence value (unless $c_1 = 1$). Similarly, abduction and exemplification can be replaced by conversion-then-deduction, but again with a reduction of confidence (when $k \geq 1$). These results show that each application of a syllogistic rule in NARS will cause some information loss (while preserving other information, of course), and therefore *direct* conclusions will always be more confident. On the other hand, the fact that exactly the same frequency value is arrived at by following different inference pathways shows that the truth-value functions defined above have not been coined individually in *ad hoc* ways, but are closely related to each other, since all of them are based on the same semantic interpretation of the truth value.

4.3 Rules for the other two inheritance relations

After determining the inference rules for the basic inheritance relation "$\subset$", it is not difficult for us to get inference rules for the two derived inheritance relation, the symmetric inheritance relation "$=$" and the singular inheritance relation "$\in$".

By definition, evidence (positive and negative) for either "$S \subset P$" or "$P \subset S$" is also evidence for "$S = P$". Therefore, weight of evidence for "$S = P$" is defined as $w^+ = |E_S \cap E_P| + |I_S \cap I_P|$, $w = |E_S \cup E_P| + |I_S \cup I_P|$. The above definition also implies that the two judgments "$S \subset P < f_1, c_1 >$" and "$P \subset S < f_2, c_2 >$" can be combined into "$S = P < f, c >$" by the revision rule defined previously (of course, the premises should be supported by non-overlapping evidence).

A similarity judgment can also be obtained by comparing the relations of two terms to a third term — for example, from premises "$P \subset M < f_1, c_1 >$" and "$S \subset M < f_2, c_2 >$" to "$S = P < f, c >$". The truth-value function of this *comparison* rule is, as before, based on the semantics of **L** and on the functions T-norm/T-conorm. The frequency of the conclusion is determined by how similar S and P are, judged from their relations to M only. Naturally, we have $f = 1 - |f_1 - f_2|$, which is the complement of the difference between f_1 and f_2. The weight of evidence of the conclusion is no bigger than 1 (analogous to the cases of induction and abduction), and is conjunctively determined by the confidence values of the premises and $S(f_1, f_2)$ (because at least one premise should be positive, analogous to the case of deduction). In summary, we have

$$\begin{aligned} F_{com}: \quad f &= 1 - |f_1 - f_2| \\ c &= [c_1 c_2 (f_1 + f_2 - f_1 f_2)]/[c_1 c_2 (f_1 + f_2 - f_1 f_2) + k] \end{aligned}$$

Using a similarity judgment as a premise, NARS can do a certain type of *analogy* — that is, it can replace a term in a judgment by a similar term to get a new judgment. For example, if the system is given the premises "$apple \subset fruit < f_1, c_1 >$" and "$pear = apple < f_2, c_2 >$", it can come to the conclusion "$pear \subset fruit < f, c >$".

It is easy to see that the situation here is quite similar to that of deduction. If $f_2 = c_2 = 1$, *pear* and *apple* become identical, therefore $< f, c >$ should be $< f_1, c_1 >$. If $f_2 = 0$, *pear* and *apple* share no common relations, therefore $f = 0$ — that is, no positive evidence for "$pear \subset fruit$" can be obtained in this way. When $f_1 = f_2 = 0$, there is no evidence for the conclusion. Exactly the same boundary condition for f applies as in deduction. Consequently, we take $f = T(f_1, f_2)/S(f_1, f_2)$.

However, as for the confidence of the conclusion, there is a subtle difference between analogy and deduction. If we change the second premise in the example into "*pear* $\subset$ *apple* $< f_2,\ c_2 >$", then it becomes deduction. Since all evidence for "*pear* $\subset$ *apple*" is evidence for "*pear* $=$ *apple*" but not vice versa, we expect the confidence for the analogical conclusion to be lower than that of the deductive conclusion (i.e., to be more sensitive to the decrease of c_2), unless $c_2 = 1$. For this reason, we take $c = T(c_2, c')$, where c' is the confidence function of the deduction rule.

In summary, for analogy we have:

$$\begin{aligned} F_{ana}: \quad f &= f_1 f_2/(f_1 + f_2 - f_1 f_2) \\ c &= c_1 c_2^2 (f_1 + f_2 - f_1 f_2) \end{aligned}$$

We also define f to be 0 when both f_1 and f_2 are 0.

A variation of analogy arises if the two premises are "*apple* $=$ *orange* $< f_1,\ c_1 >$" and "*pear* $=$ *apple* $< f_2,\ c_2 >$", with the conclusion being "*pear* $=$ *orange* $< f,\ c >$". The inference here is based on the transitivity of the "$=$" relation, and the order of the two premises does not matter. We can see it as deduction going in both directions; therefore, the truth-value function of the deduction rule is used here.

In NARS, the singular inheritance relation "$\in$" is defined in terms of the basic inheritance relation "$\subset$". Consequently, it is easy to get inference rules for the former from those for the latter. For example, since from "$\{S\} \subset M$" and "$M \subset P$" NARS can get "$\{S\} \subset P$" by deduction (treating $\{S\}$ as a term), this mode of inference is equivalent to deducing "$S \in P$" from "$S \in M$" and "$M \subset P$".

In set theory the situation is just the reverse, where "$\subset$" (subset) is defined in terms of "$\in$" (membership). NARS is not built in that way because "$\subset$", as a transitive relation, plays a fundamental role in syllogism. We cannot build syllogisms on the "$\in$" relation alone. From "$S \in M$" and "$M \in P$", we cannot get an inheritance relation between S and P. For example, from "Tweety is a robin" and "Robin is a species", we cannot establish a direct relation between "Tweety" and "species".

4.4 A summary of the rules

In the following, the rules derived in this chapter are formally displayed in several tables.

- **Revision rules**

Revision happens when the bodies of evidence of the two premises are not overlapping, and can both be used as evidence for the same conclusion. In this situation, either the premises must share two terms and must have the same inheritance relation, or else one must have an inheritance relation that is a special case of the other (e.g., "$=$" to "$\subset$").

The revision rules of NAL2 are listed in Table 4.1, where the truth values of the conclusions are all calculated by the function F_{rev} defined previously.

$J_2 \setminus J_1$	$S \subset P$	$P \subset S$	$S = P$	$S \in P$	$P \in S$
$S \subset P$	$S \subset P\ (F_{rev})$	$S = P\ (F_{rev})$	$S = P\ (F_{rev})$		
$P \subset S$	$S = P\ (F_{rev})$	$P \subset S\ (F_{rev})$	$S = P\ (F_{rev})$		
$S = P$	$S = P\ (F_{rev})$	$S = P\ (F_{rev})$	$S = P\ (F_{rev})$		
$S \in P$				$S \in P\ (F_{rev})$	
$P \in S$					$P \in S\ (F_{rev})$

Table 4.1: Revision rules.

- **Syllogisms**

The syllogistic rules take two premises that share a common term and are based on non-overlapping evidence, and from them generate an inheritance relation between the unshared terms.

Pulling together the conclusions that can be gotten by exchanging the order of the premises, we get Table 4.2 for the syllogistic rules of NAL2, where the names of truth-value functions indicate which function should be used for each case.

$J_2 \setminus J_1$	$P \subset M$	$M \subset P$	$M = P$	$P \in M$	$M \in P$
$S \subset M$	$S \subset P\ (F_{abd})$ $P \subset S\ (F'_{abd})$ $S = P\ (F_{com})$	$S \subset P\ (F_{ded})$ $P \subset S\ (F'_{exe})$	$S \subset P\ (F'_{ana})$	$P \in S\ (F'_{abd})$	
$M \subset S$	$S \subset P\ (F_{exe})$ $P \subset S\ (F'_{ded})$	$S \subset P\ (F_{ind})$ $P \subset S\ (F'_{ind})$ $S = P\ (F_{com})$	$P \subset S\ (F'_{ana})$	$P \in S\ (F'_{ded})$	
$S = M$	$P \subset S\ (F_{ana})$	$S \subset P\ (F_{ana})$	$S = P\ (F_{ded})$	$P \in S\ (F_{ana})$	$S \in P\ (F_{ana})$
$S \in M$	$S \in P\ (F_{abd})$	$S \in P\ (F_{ded})$	$S \in P\ (F'_{ana})$	$S = P\ (F_{com})$	
$M \in S$			$P \in S\ (F'_{ana})$		$S \subset P\ (F_{ind})$ $P \subset S\ (F'_{ind})$ $S = P\ (F_{com})$

Table 4.2: Syllogistic rules.

If one compares Table 4.2 with corresponding attempts in first-order predicate logic, such as (Michalski, 1993), one finds that term-oriented logics, like NAL and Peirce's logic (Peirce, 1931), provide a simpler and more natural way to combine different types of inferences.

- **Truth-value functions**

The truth-value functions appearing in Table 4.1 and Table 4.2 are listed in Table 4.3, in their "frequency–confidence" form.[5] Given the known algebraic relationships among the different forms of truth values (see Table 3.1), it is not difficult to rewrite the functions in their weight-of-evidence or frequency-interval forms (Wang, 1994c). It is sometimes possible to find direct intuitive justifications for a given algebraic function in a manner different from the way we derived them (for example, Bai Shuo in (Bai, 1991) reached the same formula for the revision function from a different starting point), but not always.

	f	c
F_{rev}	$\frac{c_1(1-c_2)f_1+c_2(1-c_1)f_2}{c_1(1-c_2)+c_2(1-c_1)}$	$\frac{c_1(1-c_2)+c_2(1-c_1)}{c_1(1-c_2)+c_2(1-c_1)+(1-c_1)(1-c_2)}$
F_{ded}	$f_1f_2/(f_1+f_2-f_1f_2)$	$c_1c_2(f_1+f_2-f_1f_2)$
F_{abd}	f_2	$f_1c_1c_2(f_1c_1c_2+k)$
F_{ind}	f_1	$f_2c_2c_1(f_2c_2c_1+k)$
F_{exe}	1	$f_1c_1f_2c_2/(f_1c_1f_2c_2+k)$
F_{com}	$1-\lvert f_1-f_2\rvert$	$[c_1c_2(f_1+f_2-f_1f_2)]/[c_1c_2(f_1+f_2-f_1f_2)+k]$
F_{ana}	$(f_1f_2)/(f_1+f_2-f_1f_2)$	$c_1c_2^2(f_1+f_2-f_1f_2)$

Table 4.3: Truth-value functions.

[5] The formulas for these functions are by no means final. They were derived from natural boundary conditions on the various types of inference, and may well be refined as a result of future research.

In terms of confidence of the conclusions, we can divide the seven truth-value functions into three groups:

1. Revision is the only rule in which the the conclusion has a higher confidence value than the premises do.

2. In deduction and analogy, the confidence values of the conclusions have 1 as their upper bound, and therefore they have corresponding rules in binary logics (where analogy degenerates into the mere substitution of identical items).

3. In abduction, induction, exemplification, and comparison, the upper bound of the conclusions' confidence values is less than 1, and depends on a system parameter, k, which is a constant, but can take different values in different systems. Consequently, these rules have no counterparts in binary logic.[6]

- **Question-related rules**

In NARS, there are three types of rule that directly deal with questions. For a given question, they decide, respectively, what is an answer, which answer is better when multiple answers have been found, and how to generate derived questions.

As was stated at the end of Chapter 3, a judgment "$S \subset P <f, c>$" is an answer to questions like "$S \subset P$", "$S \subset ?$", and "$? \subset P$". Similar rules apply to the "$=$" relation and the "$\in$" relation.

If two answers J_1 and J_2 are found for a given question Q, there are several possibilities:

1. J_1 and J_2 are about different inheritance relations, such as "$S \subset P_1$" and

[6] Defining "induction" as "reverse deduction" and "abduction" as "explanation" leads to quite different results. We will discuss such issues in Chapter 7.

"$S \subset P_2$". In this case, the judgment with a higher expectation value (defined in Section 4.1) is a better answer.

2. J_1 and J_2 are about the same inheritance relation, such as "$S \subset P$", and they are based on overlapping evidence. In this case, the judgment with a higher confidence value is a better answer.

3. J_1 and J_2 are about the same inheritance relation, and they are not based on overlapping evidence. In this case, the two judgments should be combined by the revision rule to get J_3, which has a higher confidence, so is a better answer to Q than either J_1 or J_2.

An important implication of the above rules is: there is no "final answer" to any question, though the system can choose a better answer between two candidates, and the rules of choice can be easily extended into cases involving more than two candidates. Because both expectation and confidence are real numbers in (0, 1), an answer can only be the "current best", meaning that it is better than all known answers, but cannot be the "ultimate best", meaning that there will be no better one in the future. In the next chapter, we will see how this implication influences the control strategy of the system.

What the system should do when no ready-made answer can be found for a question? Because it has insufficient resources, NARS cannot exhaustively produce all combinations of premises in order to derive a desired conclusion. Instead, *backward* inference rules are used to make the system work in a goal-directed manner.

The backward inference rules of NARS are determined by the following principle: *A question Q and a judgment J will give rise to a new question Q' if and only if an answer for Q can be derived from an answer for Q' and J, by applying a forward inference rule — that is, a syllogistic rule, as defined above.*

Given this definition, backward inference is just the inverse of forward inference, and it works by "waking up" related judgments in order to answer questions that are presented to the system (this will be explained in the next chapter). A backward-inference table can be built from the syllogism table, Table 4.2, by taking the conclusions in Table 4.2 as questions (Q), one premise (J_2) as knowledge (K), and the other premise (J_1) as the derived question. After renaming the terms and rearranging the order, we get Table 4.4, in which "P" can be either a term or a question mark.

$K \setminus Q$	$P \subset M$	$M \subset P$	$M = P$	$P \in M$	$M \in P$
$S \subset M$	$S \subset P$ $P \subset S$ $S = P$	$S \subset P$ $P \subset S$	$S \subset P$	$P \in S$	
$M \subset S$	$S \subset P$ $P \subset S$	$S \subset P$ $P \subset S$ $S = P$	$P \subset S$	$P \in S$	
$S = M$	$P \subset S$	$S \subset P$	$S = P$	$P \in S$	$S \in P$
$S \in M$	$S \in P$	$S \in P$	$S \in P$	$S = P$	
$M \in S$			$P \in S$		$S \subset P$ $P \subset S$ $S = P$

Table 4.4: Backward inference rules.

The backward-inference table turns out to be identical with the syllogism table, if we ignore the truth-value functions. This elegant symmetry reveals an implicit property of the syllogistic rules of NARS — that is, for any three judgments J_1, J_2, and J_3, if J_3 can be derived from J_1 and J_2 by a syllogistic rule, then from J_3 and J_1 the system can generate J'_2, which involves the same inheritance relation between the same two terms as does J_2 (the truth values of J_2 and J'_2 may be different). Intuitively, the three inheritance relations constitute a triangle from any two sides of which the third side can be derived. Such a property does not give rise to infinite loops in the system, because if J_3 is really derived from J_1 and J_2, it must share "serial numbers" (see the previous chapter) with each of the two, which prevents the system from taking J_3 and J_1 (or J_2) as premises in further inferences.

5

Control Mechanisms

Given the logic described in the last two chapters, we now know what conclusions *can* be inferred from a system's experience, but this does not mean that those conclusions really *will* be generated by the system.

In general, to build a reasoning system, a logic alone is not enough. A control strategy is needed to pick premises and rules for each inference step, and a storage mechanism is needed to keep track of knowledge and questions as well as of intermediate results and derived questions. Carnap calls these components "methodological", and distinguishes them from the "logical" components of a system (Carnap, 1950). A similar distinction is made by Kowalski in the formula "algorithm = logic + control" (Kowalski, 1979).

Generally speaking, the logical part of a reasoning system provides the *possibility* for a conclusion to be obtained, and the control part turns some of these possibilities into *reality*. Only in the case of a system that has sufficient knowledge (meaning it can generate all possible conclusions in finite time) and sufficient resources (meaning it has both time and space to do so) can the control part be ignored as unimportant. Obviously, NARS is not such a system.

5.1 Controlled concurrency

NARS receives two types of tasks from its environment: new knowledge (in the form of judgments) and questions. To process a piece of new knowledge means to apply some inference rule to it and to some piece of old knowledge (using them as premises) to get some new conclusion. To process a question means to match it with available knowledge to find an answer or to generate some derived questions, whose answers will in turn lead to answers of the original question.

Therefore, NARS carries out both *forward* inference — that is, from judgments to judgments — and *backward* inference — that is, from a question and a judgment to a (derived) question or an answer (provided by the judgment). Such a bidirectional flow of activity is critical to NARS. If it worked only backwards, the system could not get answers to certain questions (in the previous chapter we saw that truth-value functions are attached only to *forward* inference rules). On the other hand, the system does not possess enough resources to work forwards only (thus exhaustively generating all conclusions), so it must use questions to guide its inferences in a goal-directed manner.

We say that NARS works under insufficient time resources, or "under time pressure", firstly because the system's information-processing ability (represented by the number and speed of its processors) has an upper bound, and secondly because all tasks have time constraints attached to them, and so the system cannot process them for as long as it wishes.

How to represent a time constraint (or time pressure)? The most common way in so-called "real-time systems" is to assign a deadline — a fixed, limited amount of time — to each task. However, this type of time constraint is inappropriate for NARS, for the following reasons:

1. The system cannot depend on the environment to assign such deadlines, because the resulting requirements may exceed the system's capacity.

2. In general, the system cannot anticipate how much time it ought to spend on a task when it is accepted (from the environment) or generated (by the system itself), because that depends on future events — for example, on whether an answer is found soon, and on how many new tasks show up in the near future.

3. The concept of "deadline" implicitly assumes a step function of the utility of the answer by requesting an answer at a certain time, t — that is, an answer provided before t does not get extra credit, and an answer found after t is completely useless. Such a rigid, black-and-white attitude is not suitable for many situations.

NARS' goal is not to obtain answers of a predetermined quality, but to work as efficiently as possible when resources are in short supply; for this reason, NARS *distributes* its resources among many tasks. Consequently, the time resource given to a task is not determined by an absolute deadline, but by a relative "share", which depends both on the request from the external environment and on the internal situation of the system.

In general, there are two ways to distribute time among tasks: sequential and parallel. "Sequential" means to process the tasks one by one, and "parallel" means to have more than one task being processed at the same time. NARS processes its tasks in parallel, because that is a more flexible way to distribute resources. It should be stressed for this purpose we do not need parallel processing at the hardware level (i.e., multiple processors) — such an implementation is possible, but is not necessary for the model.

To represent the time pressure on a task, an "urgency" measurement is introduced.

Definition 15 *The* urgency *value of a task is a real number in* (0, 1]. *At any given instant, if the urgency of task t_1 is u_1 and the urgency of task t_2 is u_2, then the amounts of time resources the two tasks will receive will have the ratio $u_1 : u_2$.*

Urgency is therefore a relative rather than an absolute quantity. Knowing that $u_1 = 0.4$ tells us nothing about when task t_1 will be finished or how much time the system will spend on it. If t_1 is the only task in the system at the time, it will get all of the processing time. If there is another task t_2 with $u_2 = 0.8$, the system will spend twice as much time on t_2 as on t_1.

Intuitively, we can envision NARS as having a task pool, in which there is an urgency value attached to each task. The system processes the tasks in a time-sharing manner, meaning that the processor time is cut into fixed-size time-slices, and in each slice a single task is processed. Because NARS is a reasoning system, its processing of a task divides naturally into inference steps, one per time-slice. The system distributes its time-slices among the tasks, giving each task a number of time-slices proportional to its urgency value. As a result, urgency determines the speed of processing of that task. This picture is very similar to the "parallel terraced scan", introduced by Hofstadter (Hofstadter, 1984). "The basic image is that of many 'fingers of exploration' simultaneously feeling out various potential pathways at different speeds, thanks to the coexistence of pressures of different strengths." (Hofstadter and FARG, 1995)

If the urgencies of all tasks remain constant, then a task that arises later will get less time than a task that arises earlier, even if the two have the same urgency value. A natural solution to this difficulty is to introduce an "aging" factor for the urgency of tasks, so that all urgency values gradually *decay*.

Definition 16 *The* durability *factor (or value) of a task is a real number in* (0, 1).

If at a given moment a task has urgency value u and durability factor d, then after a certain amount of time has passed, the urgency of the task will be du.

Therefore durability is also a relative measurement. If at a certain moment $d_1 = 0.4$, $d_2 = 0.8$, and $u_1 = u_2 = 1$, we know that at this moment the two tasks will get the same amount of time resources, but when u_1 has decreased to 0.4, u_2 will only have decreased to 0.8, so the latter will then be receiving twice as much processing time as the former. (Note that under this definition, if a task has a *high* durability factor, it decays *slowly*.)

By assigning different urgency values and durability values to tasks, the environment (i.e., a human user or another computer system) can put many independent types of time pressure on the system. For example, we can inform the system that some tasks need to be worked on right now but that they have little long-term importance (by giving them high urgency values and low durability values), and that some other tasks should be processed for a longer time (by giving them high durability values).

Aging is not the only factor that changes the urgency distributions among the tasks. The amount of time spent on a task is determined not only by requirements of the environment, but also by the current result(s) the system is getting for the task. For example, if the system has found a good answer to a question, the question should become less urgent, and its durability factor should also be decreased (so it is given less time).

For these reasons, each time a task is processed (i.e., during each inference step), the system re-evaluates the task's urgency and durability values, to reflect the current situation. As a result, NARS maintains a *dynamical* distribution of urgencies in its task pool.

When should the system stop processing a task? Ideally, as in conventional computer systems, this should happen when the task has been "finished". For a piece of new knowledge, this would mean that the system had generated all its implications. For a question, it would mean that the system had found the best possible answer, given its knowledge. In both cases, the system would need to access all relevant knowledge. However, under the assumption of insufficient resources, such exhaustive use of knowledge is not possible for NARS. When time is in short supply, some pieces of knowledge have to be ignored, even if they may be relevant.

The most common method for dealing with insufficient resources is to retreat to a stance of accepting *satisfactory* solutions instead of *complete* solutions. For example, we can limit the maximum number of steps of forward inference for new knowledge, or set a threshold for the confidence or expectation of the answers. These types of methods are widely used in heuristic-search systems, but they are too inflexible for the purpose of NARS. Because the supply and demand of resources are constantly changing in NARS, such thresholds are sometimes too high (therefore the system still cannot satisfy them) and sometimes too low (therefore the system makes no attempt to get better results even though resources are still available).

The solution used in NARS is: when a task is removed from the task pool, it is not because the processing of the task has met some predetermined goal, but because the task has lost too much ground in the competition for resources. Thus, when the task pool is exceeded (it has a constant capacity), tasks at the low end of the urgency spectrum are removed. This strategy means that resource allocation in NARS is context-dependent. Even if we provide the same task, with the same urgency and durability values, to the system, it may be processed differently: when the system is busy (that is, there are many other tasks with higher urgency values), the task will be processed only briefly, and only "shallow" implications or answers will be found;

when the system is relatively idle (that is, there are few other tasks), the task will be processed more thoroughly, and "deep" results can be obtained. Generally speaking, a task can be processed for any number of steps, as in "anytime" algorithms (Boddy and Dean, 1994). The actual number of steps to be carried out is determined both by its initial urgency and durability values, and by the resources competing in the system.

If the task is a question asked by the environment, when to report an answer? In standard theories of computation, an answer gets reported only at the "final state", when the system has completed its processing of the question. However, when time is treated as a limited resource, it is reasonable to ask the system to try to provide some sort of answer as soon as possible. As was explained above, when an answer is found, it does not necessarily mean that the system will stop processing the task. Indeed, according to the question-related rules summarized in Section 4.4, it is always possible for NARS to find a better answer for a question no matter what has been found, because the quality of an answer, represented by its confidence or expectation value, can never reach 1, but can only approach it asymptotically.

As a result, the system may report more than one answer for a question — it can change its mind when new evidence is taken into account, like trial-and-error procedures (Kugel, 1986). The system keeps a record of the best answer it has found for each question, and whenever a new candidate is found, it is compared with the current best. If the new one is better (that is, has a higher confidence or expectation), it is reported to the user, and the record is updated. On the other hand, it is also possible for a question to be removed from the task pool before even a single answer is found for it.

NARS constantly generates derived tasks (judgments and questions) with its inference rules. Each such task is assigned urgency and durability values by the system, and then put into the task pool. After that, it is treated just like a task provided by the environment. Even if a “parent” task has been removed (by losing out in the competition for resources), “child” tasks derived from it may still be processed, provided that they have sufficiently high urgency values. For example, when solving a question Q, NARS may generate two derived questions Q_1 and Q_2 (by backward inference). Later, it finds an answer to Q_1, which leads to an answer to Q. At this point, the urgency values of Q and Q_1 are decreased more rapidly than that of Q_2, and it is possible for Q_2 to be processed even after Q has been removed from the system’s task pool. If the purpose of a system were solely to answer questions coming from the environment, the above strategy would seem pointless, because Q_2 is merely a means to solve Q, hence should go away if Q goes away. However, the purpose of NARS is to adapt to its environment, which means that Q_2, as a derived question, has value for its own sake, even in a situation where the question that engendered it has utterly vanished.

What I have described in this section is what I call “controlled concurrency”. Its main features, summarized, are these: the environment provides the system tasks from time to time, giving each task an urgency value and a durability factor. In each inference step, the system picks a task according to the current urgency distribution, generates new tasks as results of the interaction between that task and relevant knowledge, then adjusts the urgency of the task according to its durability value and the result of the inference. An answer is reported to the environment as long as it is the best the system has so far found for the given question. Tasks with low urgency are removed as a result of competition for resources.

This control mechanism is different from that of ordinary time-sharing, because here the tasks work on a common knowledge base, and it is not guaranteed that all tasks will be processed all the way to their final conclusions. Consequently, the interaction among tasks in NARS is much stronger and more competitive than that among the processes in a conventional time-sharing system.

5.2 Bags and chunks

The above ideas can be generalized. Let us say that a system has some items to process in a certain way. Because new items may arrive at any time, and because the time requirements of the items would exceed the system's capacity, it is impossible for the system to do the processing exhaustively. It has to distribute its time resources among the items, and to truncate the processing of an item before reaching its "final conclusion". Furthermore, items are not treated equally. The system evaluates the relative priority of each item as a function of several factors, and adjusts its evaluation when the situation changes. In addition, the system's storage capacity, which is a constant, is also in short supply.

Because this abstract phenomenon pervades the discussion of systems with insufficient resources, it will be useful to characterize it in terms of a class of *objects*, in the sense of object-oriented programming, where an object is a data structure with certain specific operations defined on it.

Let us define a class called "bag". A bag can contain a constant number of *items.* Each item has a *priority value*, which is a real number in (0, 1]. There are two valid operations defined on a bag: *put-in* and *take-out.* The operation *put-in* takes an item as argument, and has no return value. Its function is to put the item into the bag. If the bag is already full, the item with the lowest priority is first removed from it, and then the new item takes its place in the bag. The operation *take-out* has no argument, and returns one item when the bag is not empty (when the bag is empty, a special symbol is returned). The probability that any given item will be chosen, its "accessing rate", is proportional to its priority.

Now we can describe the memory structure of NARS in terms of bags. In NARS, each inference step takes a task and a piece of knowledge as premises. We know, from the description of the logical part of the model, that the two premises must share a common term. This simple property of term logic provides a natural way of focusing the selection of premises to within a small subset of the system's full knowledge. For a task with the form "$S \subset P$", the knowledge that can directly interact with it must have an S or a P in it. This suggests clustering tasks and pieces of knowledge according to the terms they contain. In particular, if we put all tasks and knowledge that share a common term together, call it a *chunk*, and name it by the shared term, then any valid inference step will necessarily happen within a single chunk.[1]

Defined in this way, a chunk becomes a higher-level unit of resource allocation, which is larger than a task or a piece of knowledge. Intuitively, a chunk corresponds to a concept. The name of a chunk is a *term*, and the body of the chunk contains the *meaning* of that term — that is (according to the experience-grounded semantics introduced in Chapter 3), its experienced relations with other terms. The memory of the system is simply a set of chunks.

Now the action of choosing a task can be recast as a two-step process: first choosing a chunk, and then from it choosing a task. In other words, the system distributes its resources firstly among the chunks, and then secondly, within each chunk, among the tasks. The result is a two-level structure. On both levels, the notion of "bag" applies. Specifically, we can describe the memory of NARS as a bag of chunks, with, within each chunk, a bag of tasks.

Let us assume that the system has picked some chunk, S, and a task inside it,

[1] Obviously, such an approach introduces a certain amount of redundancy into the system. For example, the task "$S \subset P$" belongs to both chunk S and chunk P.

"$S \subset P \ <f, c>$", which is a piece of new knowledge. Now how to choose another piece of knowledge to be used with it? As was stated above, the other piece of knowledge should also come from within chunk S. There are five types of knowledge in the chunk, of the forms "$S \subset x$", "$x \subset S$", "$S \in x$", "$x \in S$", and "$S = x$" respectively, where x can be any term.

From Table 4.2 we can see that given the type of the task, different types of knowledge lead to different types of inference. In the current example, the inference may be deduction (if "$x \subset S$" or "$x \in S$" is chosen), induction and comparison (if "$S \subset x$" is chosen), analogy (if "$S = x$" is chosen), or even nothing (if "$S \in x$" is chosen). Since the purpose of forward inference is to reveal the implications of new knowledge, the system tends to prefer deductive conclusions to inductive conclusions (because the former usually have higher confidence), and will not pick a piece of knowledge that leads to no conclusion for this step. For each type of knowledge, the system prefers items that have higher confidence and that are more relevant to the current situation. Therefore, knowledge choosing is also a two-step action. First, a knowledge type is chosen probabilistically as a function of the type of the task, and then a piece of knowledge of the chosen type is chosen. The second step is also carried with the aid of the "bag" notion defined above. In particular, the pieces of knowledge in each chunk are organized by *type*, so that each chunk contains five knowledge bags.

In summary, the memory of NARS is a bag of chunks, with each chunk consisting of a task bag and five knowledge bags. To make the discussions easier, in the following I will use *activity*, *urgency*, and *importance* for the priority value of a chunk, a task, and a piece of knowledge, respectively.

Now we can see the distinction between *tasks* and *pieces of knowledge* more clearly. All questions are tasks. All judgments are knowledge. New knowledge also serves as tasks for a short time. If a piece of knowledge provides an answer for a question, it will be treated as a task for a short time. Because of this distinction, the system has, at any given moment: (1) a small number of tasks, which are active, remembered for a short time, and highly relevant to the current situation; and (2) a much larger amount of knowledge, which is passive, remembered for a long time, and mostly not relevant to the current situation.

5.3 An atomic step

Processing in NARS consists of the repeated execution of the following sequence of operations, which constitutes one "step of inference". Such a step always takes a (nearly) constant amount of time, no matter how big the memory is. This is a consequence of the fact that all operations listed below take roughly constant time.

(1) Handling of inputs

In principle, the environment can provide new questions and knowledge to the system at any time. We can assume that the inputs are placed in a buffer and are processed only at the beginning of each step. All the inputs are treated as tasks by the system. The environment provides urgency and durability values for each piece of input. The system sorts the input into the task bags of the appropriate chunks, and increases the activity of these chunks. If an input is a piece of knowledge (not a question), it will also be put into the knowledge bag of the appropriate chunks.

(2) Choice of a chunk

The system picks a chunk from the memory. As was stated earlier, the choice is probabilistic, with relative probabilities determined by the activity distribution of the chunks in the memory. Generally speaking, a chunk containing more urgent tasks is more active — that is, has a higher chance of being chosen.

(3) Choice of a task

A task is removed from the task bag of the chunk, with relative probabilities determined by the urgency distribution of the tasks in the task bag.

(4) Choice of a knowledge bag

The system makes a choice among the five knowledge bags inside the chosen chunk, again probabilistically, biased by the type of the task. Generally speaking, a knowledge type that usually leads to better results has a higher chance.

(5) Choice of a piece of knowledge

A piece of knowledge is taken out of the chosen knowledge bag, with relative probabilities determined by the importance distribution of the pieces of knowledge in the knowledge bag.

(6) Inference-drawing

Unlike many other systems, NARS does not decide what type of inference will be used to process a task at the moment the task is specified, but works in a *knowledge-driven* way — that is, it is the combination of task and knowledge that happen to be picked that determines what type of inference will be carried out. Depending on the combination of task and knowledge, the inference will be one of the following:

match: if the task is a question, and the piece of knowledge happens to be the current best answer to the question, a copy of the piece of knowledge is generated as a new task, and also reported to the user if the question is an input question (i.e., asked by the user);

backward inference: if the task is a question, the piece of knowledge may lead to derived questions (Table 4.4);

revision: if the task and the piece of knowledge happen to provide evidence for the same inheritance relation, the evidence is accumulated (Table 4.1);

forward inference: if the task and the piece of knowledge provide evidence for two inheritance relations that share exactly one term, new relations are formed between the unshared terms (Table 4.2).

The making of the inference thus results in one or more derived tasks.

(7) Evaluation of priorities

The system needs to assign urgency and durability values for the derived tasks, and to adjust the urgency/importance and durability value of the task/knowledge that are serving as premises in the current step, and after doing so, to put them back into the their respective bags. After that, the activity and durability values of the current chunk are also adjusted, and the chunk is returned into memory.

(8) Handling of results

The derived tasks are placed in the input buffer and will be processed, like input tasks, at the beginning of the next atomic step.

5.4 Priority and durability

What has not been discussed is the subtlest part of the control mechanism: how to determine the priority and durability values for chunks, tasks, and pieces of knowledge in phase (7) of each atomic step. To this end, we need to design a set of numerical functions.

Since all the quantities involved are real numbers in [0, 1], these functions have been designed in a manner reminiscent of the design of the truth-value functions.

First, the boundary condition constraining the desired function is determined, and then a function satisfying this condition is formed with the help of T-norm, T-conorm, and other general-purpose operators (such as difference, average, and so on). Functions obtained in this way are by no means optimal. They only reflect my current analysis of the situation, and my ideas are still being revised from time to time. In the following, therefore, we will discuss them only qualitatively.

- **Tasks**

The urgency and durability values of an input task are assigned by the user, and reflect the time constraint placed by the user on the task. Of course, the system can provide default values.

The urgency and durability values of a *derived* task are determined by several factors: generally, it inherits them from its parents. If both the task and knowledge that derive the new task have high priority (or durability) values, the new task will get a high urgency (or durability) value. However, the values will also be affected by the "quality" of the inference result. In revision, a new task gets a high urgency value if the two premises have significantly different truth values (a belief conflict). In forward inference, a new task gets a high urgency and durability values if it is a confident conclusion. In backward inference, if the answer to the derived task can (with its parent knowledge) generate a good answer to the parent task, it gets high urgency and durability values. If the given task is a question and the given piece of knowledge provides an answer for it, the urgency and durability values of the new task (which is a copy of the piece of knowledge) will depend on how good the answer is.

After the inference, the urgency value of the parent task is decreased. In addition to the decrease due to the durability factor described earlier, there may be further decrease, depending on the quality of the result. If an answer to a question is found and is quite good (that is, has a high confidence or expectation), then the urgency of the question is decreased more than if the answer is poor.

- **Knowledge**

Initially, a piece of knowledge is generated by copying some task. It is given the highest importance value (that is, 1), and its durability value is determined by the urgency and durability values of that task. In this way, new knowledge gets more attention from the system, but usually decays rapidly unless it corresponds to an urgent and long-term task.

After a piece of knowledge is used, its importance and durability values are adjusted. In contrast to the case of tasks, the importance value of a piece of knowledge can be increased. In each adjustment, there are two competing forces pulling the importance value in opposite directions. One force is the universal aging force, which decreases the importance value by multiplying it by the durability factor (a number less than 1). On the other hand, the piece of knowledge gets rewarded each time (by increasing its importance value) depending on how good the result is in this step. Therefore, in the long run, useful knowledge (which has provided good answers and implications in the past) gets high importance values, which makes it more accessible in the future.

The durability value of any given piece of knowledge is increased each time. As a result, the longer a piece of knowledge stays in the memory, the more slowly its importance value will decrease.

- **Chunks**

The activity value of a chunk increases when a task is put into it, with the size of the increase depending on the urgency value of the task. At the same time, the chunk's durability value is also adjusted. If the chunk already has high activity before the task is inserted into it, it will be given a high durability value, which ensures that the current high activity will be kept longer. On the other hand, if the chunk's activity took a big, upwards jump, then the durability value will be set low, so the chunk will lose its activity soon.

The activity value of a chunk decreases whenever any task is removed from it. In addition to the decrease due to the durability factor, the size of the decrease also depends on the urgency of the task that was just removed. Specifically, if the task had a high urgency value, the decrease will be larger.

- **Summary**

What makes the above functions different from the heuristics used in expert systems is their domain-independent nature. In the design of these functions, no assumption was made about the *content* of the task or the knowledge. However, this does not mean that NARS uses a general-purpose, context-independent algorithm, which always uses the same predetermined method to solve problems, and is insensitive to available domain knowledge. Indeed, there is a pervasive influence of domain knowledge on the control of inference-making at all times when the system is running. As was laid out above in great detail, the system's choice at each step strongly depends on the available and activated domain knowledge, which is certainly not predetermined by the designer. On the other hand, the *design* of the system, including all the functions discussed above, is independent of any particular domain.

Another property worth mentioning is that in each step, only those chunks, tasks, and pieces of knowledge that are directly involved have their priority and durability values adjusted. Thus, no matter how big the memory is, this set of adjustments takes roughly constant time. This is exactly what one would wish for. On the other hand, these adjustments have global effects. Since priority and durability are *relative* values, the increase of the urgency of any particular task means that all competing tasks will have less chance. Though such effects are small at each step, they accumulate in the long run, and result in a dynamic distribution of resources.

With all these control-related quantities adjusted dynamically, some tasks are processed faster, and some pieces of knowledge are more accessible, while others slowly slide into dormancy. And because storage space is limited, chunks, tasks and pieces of knowledge with sufficiently low priorities may wind up being permanently forgotten. This, though sometimes disadvantageous from a practical point of view, is the price any system has to pay when its knowledge and resources are insufficient.

6

Computer System

The first version (1.0) of NARS was implemented in 1986, for my Master of Science degree. The programming language was Prolog. The results are presented in (Wang, 1986; Wang and Hsu, 1987).

In 1992 and 1993, in preparation for my Ph.D. project, I implemented versions 2.0, 2.1, and 2.2, all in Scheme (Wang, 1993c; Wang, 1993d).

In this chapter, the most recent implementation of the NARS model, version 3.0, will be described.

6.1 Internal structure

From the descriptions of the model in the previous chapters, we can see that NARS can be cast quite naturally in the framework of object-oriented programming. For this reason, NARS 3.0 was written in C++.

The kernel of the implementation is the "bag" class. As was described previously, the "take-out" operator makes its selection by looking at the priority distribution of items within the bag. A very precise implementation of this behavior, though not impossible, is not necessary for the current purpose. What we need most of all is a good approximation to the ideal behavior, plus adherence to the constraint that the take-out operator (as well as the put-in operator) should take a roughly constant time, preferably a very short time.

The actual implementation divides the items in a bag into several levels, according to their priority. Thus a continuum of values is approximated by a discrete set of levels. The take-out operator visits these levels according to a predetermined probability distribution. Within each level, items are removed in a "first-in-first-out" order. The put-in operator inserts a new item into the level that corresponds most closely to the item's priority. When a bag is full and a new item must be inserted, an item at the lowest non-empty level is removed. In this way, items are probabilistically accessed in a manner that closely reflects their priority, and yet their priorities are never directly compared with one another; this design increases the computational efficiency of the operations.

Using the "class inheritance" mechanism provided by C++ (which should not be confused with the "inheritance relation" between terms in NARS), three more specific classes are defined as specializations of the "bag" class: "chunk-bag", "task-bag", and "knowledge-bag". To reduce the computational overhead, what is actually found in the "chunk-bag" is not chunks themselves (with their own task bags and knowledge bags inside them), but the *names* of chunks. Once a name has been selected, a look-up table is used to get the actual address of the chunk from its name. Furthermore, the "chunk-bag" is divided into two bags, one for "active chunks", the other for "dormant chunks"; both of these sub-bags have constant space, but only the *active* chunks participate in the competition for time resources. When the bag of active chunks is full, a name at the lowest non-empty level is moved into the bag for dormant chunks, and thus that chunk is withdrawn from the competition for processor time. Of course, any dormant chunk can be activated by the arrival of a new task. When the dormant-chunk bag is full, some chunk is erased from the system. Therefore, active chunks are given (and fight over) both time and space resources, whereas dormant chunks are given (and fight over) space resources only. In future implementations of NARS, we can expect a relatively small active-chunk bag and a huge dormant-chunk bag. In the current implementation, however, both bags are small.

Aside from these bags, other major components of the system include the inference tables (see Section 4.4), the truth-value functions (also see Section 4.4), and the priority/durability functions (see Section 5.4).

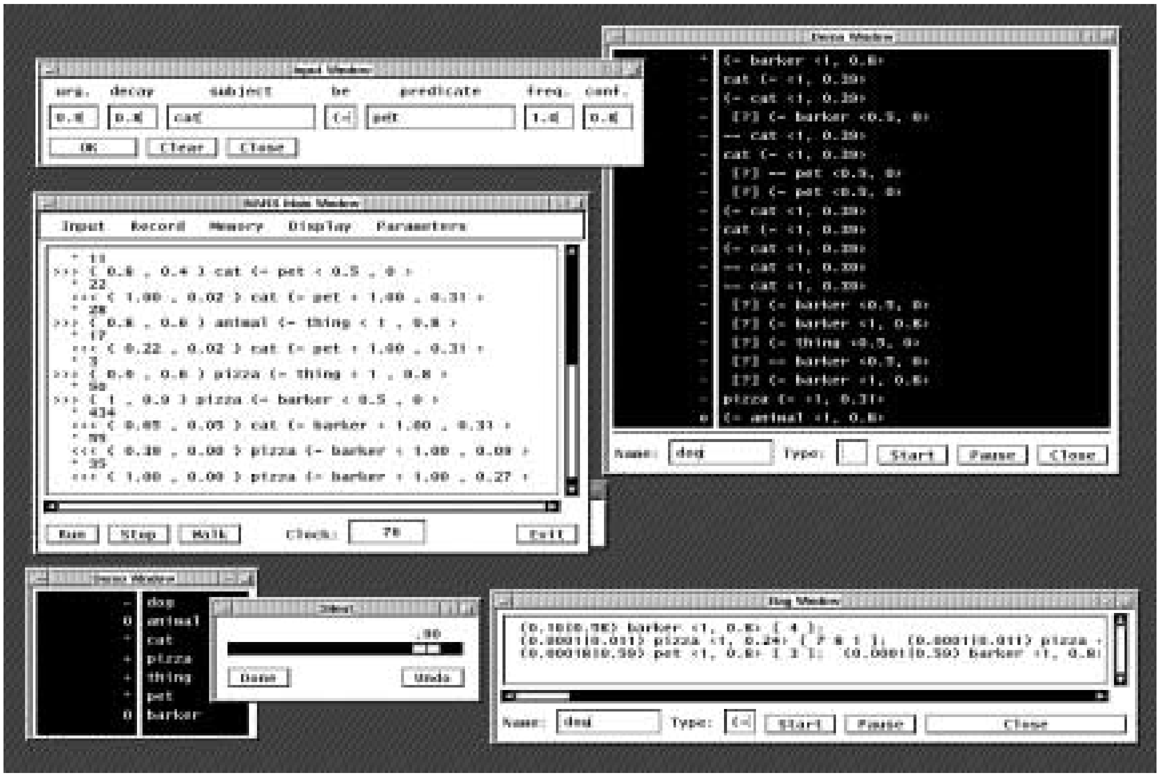
There are two types of parameters that can be tuned. The first type includes all space specifications, such as the sizes of various bags. These parameters are defined as constants, meaning that they can be changed before compilation but cannot be adjusted when the system is running. The second type includes the aging rates for chunks, tasks, and knowledge, which can be adjusted when the system is running. As defined in Chapter 5, the durability value of an item indicates how much of its priority will remain after a specific constant amount time. The aging rates determine that time constant. The smaller any aging rate is, the smaller the corresponding constant is, therefore the faster the corresponding priority decreases.

6.2 User interface

The user interface of NARS 3.0 has been developed using the OSF/Motif toolkit, which works smoothly with C++ classes.

A nice mechanism provided in Motif is “work procedure”, which is a background routine that is repeatedly called whenever no interface events happens. The atomic inference step of NARS (Section 5.3) exactly fits into such a work procedure. Implemented in this way, the system responds to user operations immediately, and the expenses and complexity of implementing multiple processes are avoided.

Figure 6.1 shows a screen snapshot when the system is working on an example (Example 5 of the next section). The other window snapshots in the following are also taken from the same example, at different moments. All these windows except the main window can be created and destroyed interactively by the user.



urg. 0.8, decay 0.8, subject cat, be <-, predicate pet, freq. 1.0, conf. 0.8

```
  * 11
>>> [ 0.8 , 0.4 ] cat <- pet < 0.5 , 0 >
  * 22
  <<< < 1.00 , 0.02 > cat <- pet < 1.00 , 0.31 >
  * 28
>>> [ 0.8 , 0.6 ] animal <- thing < 1 , 0.8 >
  * 17
  <<< < 0.22 , 0.02 > cat <- pet < 1.00 , 0.31 >
  * 3
>>> [ 0.8 , 0.8 ] pizza <- thing < 1 , 0.8 >
  * 58
>>> [ 1 , 0.9 ] pizza <- barker < 0.5 , 0 >
  * 434
  <<< < 0.85 , 0.05 > cat <- barker < 1.00 , 0.31 >
  * 99
  <<< < 0.38 , 0.00 > pizza <- barker < 1.00 , 0.08 >
  * 35
  <<< < 1.00 , 0.00 > pizza <- barker < 1.00 , 0.27 >
```

Clock: 78

```
+ <- barker <1, 0.8>
- cat <- <1, 0.39>
- <- cat <1, 0.39>
-  [?] <- barker <0.5, 0>
- -- cat <1, 0.39>
- cat <- <1, 0.39>
-  [?] -- pet <0.5, 0>
-  [?] <- pet <0.5, 0>
- <- cat <1, 0.39>
- cat <- <1, 0.39>
- <- cat <1, 0.39>
- -- cat <1, 0.39>
- -- cat <1, 0.39>
-  [?] <- barker <0.5, 0>
-  [?] <- barker <1, 0.8>
-  [?] <- thing <0.5, 0>
-  [?] -- barker <0.5, 0>
-  [?] <- barker <1, 0.8>
- pizza <- <1, 0.31>
0 <- animal <1, 0.8>
```

Name: dog Type:

```
- dog
0 animal
+ cat
+ pizza
+ thing
+ pet
0 barker
```

.90

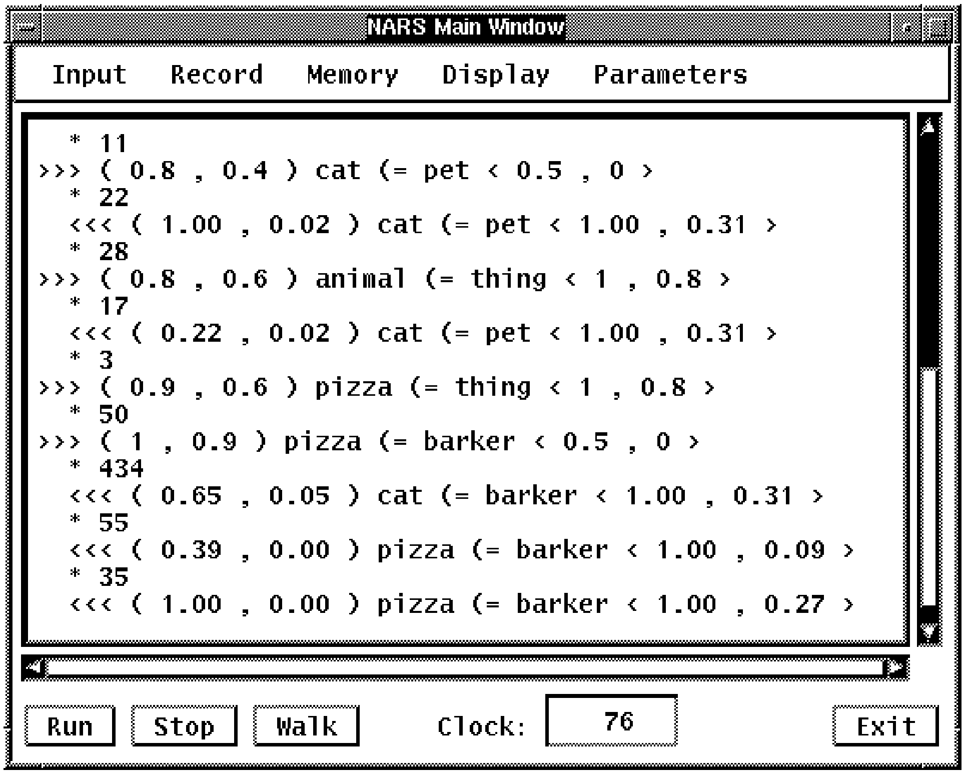
```
(0.18|0.58) barker <1, 0.8> [ 4 ];
(0.0001|0.011) pizza <1, 0.24> [ 7 8 1 ];  (0.0001|0.011) pizza <
(0.0001|0.59) pet <1, 0.8> [ 3 ];  (0.0001|0.59) barker <1, 0.8>
```

Name: dog Type: <-

Figure 6.1: Screen snapshot of NARS 3.0.

The main window of the system is shown in Figure 6.2.

The text region of the main window records the communications between the system and the user. Lines that start with “>>>” are input to the system; lines that start with “<<<” are output from the system; and the number after a “*”



```
  * 11
>>> ( 0.8 , 0.4 ) cat (= pet < 0.5 , 0 >
  * 22
  <<< ( 1.00 , 0.02 ) cat (= pet < 1.00 , 0.31 >
  * 28
>>> ( 0.8 , 0.6 ) animal (= thing < 1 , 0.8 >
  * 17
  <<< ( 0.22 , 0.02 ) cat (= pet < 1.00 , 0.31 >
  * 3
>>> ( 0.9 , 0.6 ) pizza (= thing < 1 , 0.8 >
  * 50
>>> ( 1 , 0.9 ) pizza (= barker < 0.5 , 0 >
  * 434
  <<< ( 0.65 , 0.05 ) cat (= barker < 1.00 , 0.31 >
  * 55
  <<< ( 0.39 , 0.00 ) pizza (= barker < 1.00 , 0.09 >
  * 35
  <<< ( 1.00 , 0.00 ) pizza (= barker < 1.00 , 0.27 >
```

Figure 6.2: Main window.

indicates the time interval between successive lines, in terms of inference steps. Each line of input and output shows a task, preceded by its urgency and durability values. The truth values of all judgments are represented in the $<frequency, confidence>$ format,[1] and any input task with confidence value 0 is taken to be a question (its frequency value does not matter). To indicate the various types of inheritance relations, "$\subset$" is written in ASCII characters as "(=", "$\in$" as "{−", and "=" as "==".

The three buttons on the bottom left side of the window correspond to the commands for "repeatedly execute atomic steps", "stop executing atomic steps", and "execute a given number of atomic steps", respectively. The "clock" always shows

[1] For the sake of simplicity, the "weight" and "interval" forms of truth value are not implemented in NARS 3.0.

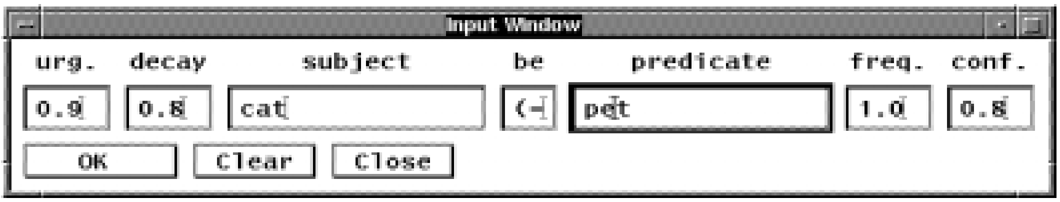
the number of steps that have been taken since the last input or output event. The button at the bottom right corner is for exiting the system.

The menu bar above the text region provides five sets of commands.

- **Input**

The system can get input either from a window or from a file.

The "open window" command will display an auxiliary window (Figure 6.3) for the user to provide questions and pieces of knowledge to the system. While the user is creating a new task, the system may still be doing inference.



urg.	decay	subject	be	predicate	freq.	conf.
0.9	0.8	cat	<-	pet	1.0	0.8

Figure 6.3: Input window.

The file-related input commands include ones to do the following: to open a specific file, to pause while reading, to continue reading, and to stop reading (and close the file).

Having the system read an "experience" file is a very efficient way to put tasks into the system, especially when the system is still under development. An experience file has precisely the same format as the text region of the main window. When the system reads such a file, lines starting with ">>>" are processed as new tasks, lines starting with "*" are interpreted as instructions for the system to take that number of inference steps, and lines starting with "<<<" are ignored. While reading from a file, the system still responds to interface events, including input from the input window.

The following is a sample experience file:

```
>>> ( 0.7 , 0.6 ) Tweety {- bird < 1 , 0.8 >
>>> ( 0.7 , 0.6 ) bird (= animal < 1 , 0.8 >
  * 10
>>> ( 0.8 , 0.4 ) ? {- animal < 0.5 , 0 >
  * 10
>>> ( 0.8 , 0.8 ) animal == plant < 0.1 , 0.8 >
  * 20
>>> ( 0.8 , 0.4 ) Tweety {- plant < 0.5 , 0 >
  * 20
```

Intuitively, here the user first tells the system that “Tweety is a bird” and that “A bird is an animal”, with the same confidence values. Since urgency and durability are defined relatively, their precise values make no difference when the two pieces of knowledge are given the same values. After being given a short time to process the knowledge, the system is asked to find an instance of “animal”. Comparing this task with the previous two, we can see that the question is given a higher urgency value and a lower durability value, so that it will get more attention at the beginning, but will be forgotten earlier than the knowledge. Again, the precise urgency and durability values are not important for the current discussion.

- **Recording of conversation**

There are commands that tell the system to start and to stop recording its experience/response (the conversation between the system and the user), respectively. With these commands, the user can store an arbitrary section of the conversation into a file, which can be used later for analyzing the system’s behavior, or can be read later by the system (or other systems) as input.

An example of a recorded conversation, with the previous experience file as input, will be discussed in the next section as Example 1.

- **Memory**

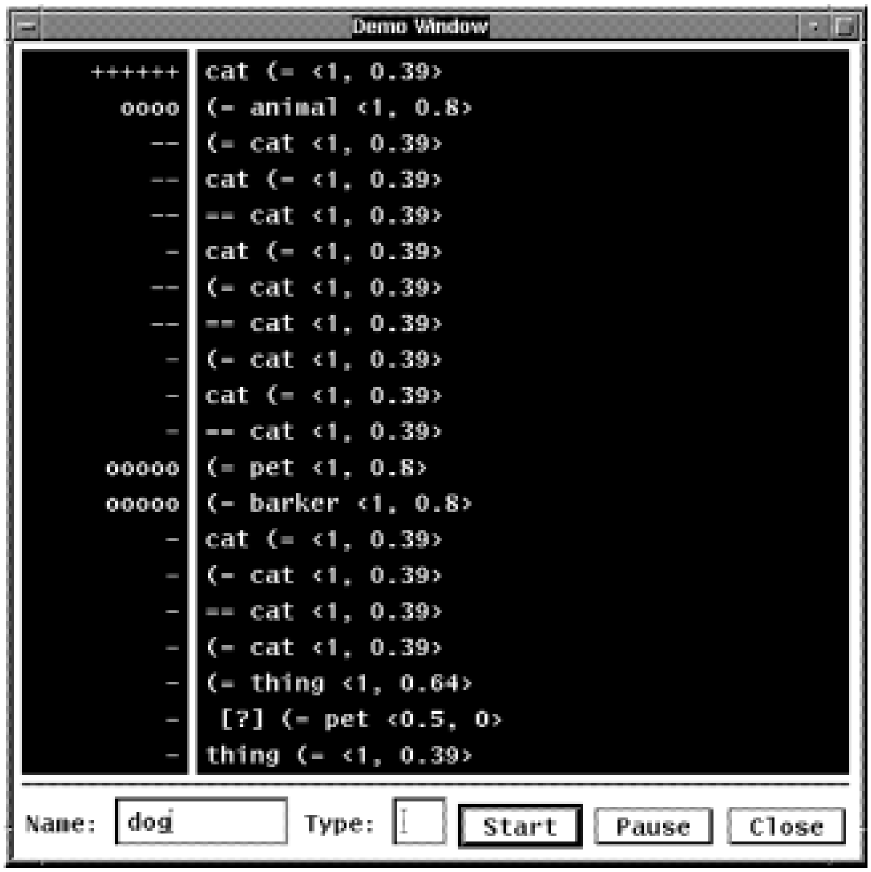
The user can save the current state of the system, including all the bags and the values of all variables, into a file. Later, such a file can be loaded into the system, to let it continue from a previous state. In this way, a user can let the system run for an arbitrarily long time in principle, even though physically the system has been "frozen" and "thawed" from time to time; such pauses have no influence on the system's behavior.

- **Display of specific bags**

There are commands to open auxiliary windows to show the content and priority distribution of a selected bag. As was stated in the previous chapter, a bag may contain the names of all active chunks, all tasks in a chunk, or all knowledge of a certain type in a chunk. For the latter two cases, the name of the chunk and the type of the bag need to be specified. The user can open several such windows to follow the changes that happen at different levels and points in the system. This function is especially useful for the tuning and debugging of the system, as well as when the system is used as a cognitive model. The "top items" option opens a "Demo window" (Figure 6.4), which displays only items with top priority values in the bag, with symbols indicating their priority and durability values. This type of window is better for dynamical display. By contrast, the "all items" option opens a "Bag window" (Figure 6.5) which contains more complete information about the contents of the bag, and is better for static analysis.

- **Parameter adjustment**

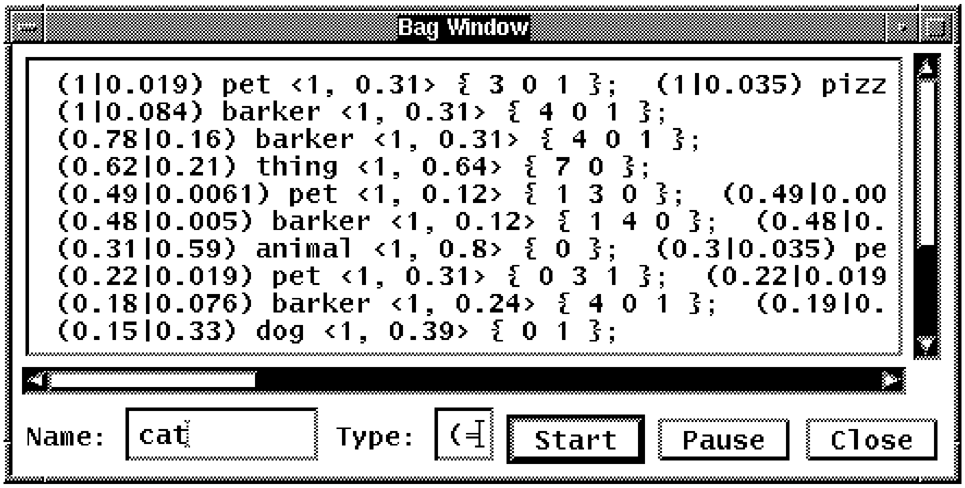
All dynamical parameters can be changed using a scroll bar, and the change can be



```
++++++ cat (= <1, 0.39>
  oooo (= animal <1, 0.8>
    -- (= cat <1, 0.39>
    -- cat (= <1, 0.39>
    -- == cat <1, 0.39>
     - cat (= <1, 0.39>
    -- (= cat <1, 0.39>
    -- == cat <1, 0.39>
     - (= cat <1, 0.39>
     - cat (= <1, 0.39>
     - == cat <1, 0.39>
 ooooo (= pet <1, 0.8>
 ooooo (= barker <1, 0.8>
     - cat (= <1, 0.39>
     - (= cat <1, 0.39>
     - == cat <1, 0.39>
     - (= cat <1, 0.39>
     - (= thing <1, 0.64>
     -  [?] (= pet <0.5, 0>
     - thing (= <1, 0.39>
```

Name: dog Type:

Figure 6.4: Demonstration window.



```
(1|0.019) pet <1, 0.31> { 3 0 1 };  (1|0.035) pizz
(1|0.084) barker <1, 0.31> { 4 0 1 };
(0.78|0.16) barker <1, 0.31> { 4 0 1 };
(0.62|0.21) thing <1, 0.64> { 7 0 };
(0.49|0.0061) pet <1, 0.12> { 1 3 0 };  (0.49|0.00
(0.48|0.005) barker <1, 0.12> { 1 4 0 };  (0.48|0.
(0.31|0.59) animal <1, 0.8> { 0 };  (0.3|0.035) pe
(0.22|0.019) pet <1, 0.31> { 0 3 1 };  (0.22|0.019
(0.18|0.076) barker <1, 0.24> { 4 0 1 };  (0.19|0.
(0.15|0.33) dog <1, 0.39> { 0 1 };
```

Name: cat Type: (=

Figure 6.5: Bag contents window.

saved or canceled (Figure 6.6). In the current version of the system, these parameters include the aging rates of chunks, tasks, and knowledge, introduced previously. There is also a threshold-like parameter that controls how "silent" the system is. When this threshold is 0, the system reports (in the text region of the main window) all conclusions it gets, no matter how trivial they are. With the increase of the threshold, the system will only report important results (that is, with high priority and durability values). When the threshold is 1, the system reports only the best answers to input questions, and keeps all other results to itself.

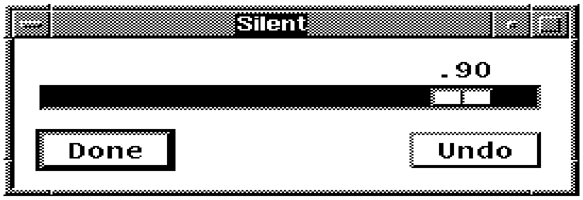


Figure 6.6: Parameter window.

6.3 Examples

In this section, we shall see how NARS works by discussing several examples that have been produced by the system. To simplify the discussion, in the following all conversations were recorded when the system had just been "born" — that is, when it has no previous experience. Also, the system is given no innate domain knowledge. All the input judgments are assigned a confidence value 0.8 by the user. Under the presumption that $k = 1$ (the constant defining the "near future", defined in Chapter 3), this means that all input judgments are supported by evidence with $w = 4$ (so $c = w/(w + k) = 0.8$).

- **Example 1**

```
>>> ( 0.7 , 0.6 ) Tweety {- bird < 1 , 0.8 >
>>> ( 0.7 , 0.6 ) bird (= animal < 1 , 0.8 >
  * 10
>>> ( 0.8 , 0.4 ) ? {- animal < 0.5 , 0 >
  * 10
>>> ( 0.8 , 0.8 ) animal == plant < 0.1 , 0.8 >
  * 4
  <<< ( 0.81 , 0.21 ) Tweety {- animal < 1.00 , 0.64 >
  * 16
>>> ( 0.8 , 0.4 ) Tweety {- plant < 0.5 , 0 >
  * 5
  <<< ( 1.00 , 0.02 ) Tweety {- plant < 0.10 , 0.41 >
```

After being told that “Tweety is a bird” and “A bird is an animal”, the system can find a single instance, “Tweety”, for “animal”, by deduction (see Section 4.4). Because all available evidence is positive, the answer is “Yes”. However, since the confidence of the conclusion is lower than either of the premises’, it will be easier to revise in the light of future evidence. Generally speaking, all syllogistic inferences cause confidence loss — that is, the confidence of the conclusion is always lower than the confidence value of both premises.

The next thing that happens is that the system is told that “animal” and “plant” are not similar to each other. By analogy, it concludes that it is very unlikely for “Tweety” to be a “plant”.

If the user changes the control factors in an experience file, such as the urgency and durability values of input tasks or the time interval between successive input tasks, the system's response changes, too, in the sense that the timings and orders of reports are usually different. However, what answers are possible is independent of these control factors, being completely determined by the system's logic.

- **Example 2**

```
>>> ( 0.7 , 0.5 ) swan (= bird < 1 , 0.8 >
>>> ( 0.7 , 0.5 ) swan (= swimmer < 1 , 0.8 >
  * 10
>>> ( 0.7 , 0.5 ) dove (= bird < 1 , 0.8 >
>>> ( 0.7 , 0.5 ) dove (= swimmer < 0 , 0.8 >
  * 60
>>> ( 0.8 , 0.5 ) penguin (= bird < 1 , 0.8 >
>>> ( 0.8 , 0.5 ) penguin (= swimmer < 1 , 0.8 >
  * 100
>>> ( 1 , 0.9 ) swimmer (= bird < 0.5 , 0 >
>>> ( 1 , 0.9 ) bird (= swimmer < 0.5 , 0 >
  * 83
  <<< ( 0.42 , 0.08 ) swimmer (= bird < 1.00 , 0.39 >
  * 2
  <<< ( 0.39 , 0.18 ) bird (= swimmer < 0.67 , 0.66 >
  * 35
  <<< ( 0.15 , 0.15 ) swimmer (= bird < 1.00 , 0.56 >
  * 4
  <<< ( 0.80 , 0.20 ) bird (= swimmer < 0.67 , 0.66 >
```

The system is told that “dove”, “swan”, and “penguin” are instances of “bird”, that “dove” is not a “swimmer”, and that “swan” and “penguin” are instances of “swimmer”. Then it is asked, “Is a swimmer a bird?” and “Is a bird a swimmer?”

According to the induction rule (see Section 4.2), “swan” and “penguin” provide positive evidence for both inductive conclusions. “Dove” is a piece of negative evidence for “A bird is a swimmer”, but has nothing to do with “A swimmer is a bird”. Each piece of evidence generates a pair of inductive conclusions for the questions, and then the corresponding conclusions are merged by the revision rule (see Section 4.1) to get summarized conclusions. As a result, “A swimmer is a bird” gets a higher frequency (no negative evidence) and a lower confidence (fewer examples) than “A bird is a swimmer”.

Since the question “Is a bird a swimmer?” is sent to both the chunk “swimmer” and the chunk “bird”, and since the two chunks work independently, the answer “$bird \subset swimmer < 0.67, 0.66 >$” is reported twice.

For the question “Is a swimmer a bird?”, the first answer is based on partial knowledge — one piece of positive evidence. Later, the other piece of positive evidence is taken into consideration, and the system provides a different answer. In this sense, NARS is a “non-monotonic logic”, though quite different from other logic systems under that name (Reiter, 1987).

From this example we can see that though the answers can be understood as statistical conclusions (such as “Three types of bird are known, of which two are swimmers”), they are not generated by looking for birds in the knowledge base, then counting swimmers among them. Instead, the system does induction from each piece of evidence, then merges the results. In this way, the producing of an inductive answer consists of many steps, and the system may stop anywhere, as a function of the real-time pressures involving its computational resources. The system may happen to find and take into account *all* relevant evidence, as in the current example, but such completeness is not guaranteed and is not typical for more complex examples.

- **Example 3**

```
>>> ( 0.6 , 0.5 ) sport (= physical-activity < 1 , 0.8 >
>>> ( 0.6 , 0.5 ) chess (= physical-activity < 0.2 , 0.8 >
  * 5
>>> ( 1 , 0.9 ) chess (= sport < 0.5 , 0 >
  * 1
  <<< ( 0.66 , 0.05 ) chess (= sport < 0.20 , 0.39 >
  * 1
  <<< ( 0.36 , 0.05 ) chess (= sport < 0.20 , 0.39 >
  * 8
>>> ( 0.6 , 0.5 ) sport (= competition < 1 , 0.8 >
>>> ( 0.6 , 0.5 ) chess (= competition < 1 , 0.8 >
  * 184
  <<< ( 0.75 , 0.12 ) chess (= sport < 0.60 , 0.56 >
```

This time, the system uses abduction to decide whether “chess” is a “sport”. Given the knowledge that “sport” usually means “physical activity” and “chess” is seldom referred to as a “physical-activity”, “chess” is judged as not a “sport”. However, the system then learns that “sport” also means “competition”, and that “chess” is a “competition”, so in this sense, “chess” is a “sport”. Finally, the positive and negative evidence is combined by the revision rule and a summarized answer is reported to the user, because the initial question, even though already answered once, is still remembered by the system.

What makes this example different from the previous one is that the answer should not be understood extensionally as “There are many types of chess, but only 60% of them are sports”. As was explained earlier, the extent of extensional inclusion is only one particular way by which the truth value of a judgment is determined. Generally, a truth value can be generated by comparing the *extensions* of two concepts (as in the earlier example) or by comparing the *intensions* of two concepts (as in the current example). These two factors can even be mixed when an extensional conclusion and an intensional conclusion are merged by the revision rule, thus becoming indistinguishable in the result. Therefore, one can only say that the truth value of a judgment records the extent to which one term "can be used as" the other, according to the system's experience, but one usually cannot tell *how* such a value was arrived at without examining the system's complete history.

- **Example 4**

```
>>> ( 0.9 , 0.6 ) fruit (= plant < 1 , 0.8 >
>>> ( 1 , 0.9 ) pear (= plant < 0.5 , 0 >
  * 5
  <<<  ( 0.52 , 0.53 ) fruit == pear < 0.50 , 0.00 >
  <<<  ( 0.52 , 0.53 ) pear (= fruit < 0.50 , 0.00 >
  * 7
  <<<  ( 0.50 , 0.53 ) fruit == pear < 0.50 , 0.00 >
  <<<  ( 0.50 , 0.53 ) pear (= fruit < 0.50 , 0.00 >
  * 8
>>> ( 0.9 , 0.6 ) apple (= fruit < 1 , 0.8 >
>>> ( 0.9 , 0.6 ) orange (= fruit < 1 , 0.8 >
>>> ( 0.9 , 0.6 ) pear == apple < 0.9 , 0.8 >
  * 5
  <<<  ( 0.77 , 0.18 ) pear (= fruit < 0.90 , 0.51 >
  * 86
  <<< ( 1.00 , 0.03 ) pear (= plant < 0.90 , 0.41 >
```

In the previous examples, the "silence" parameter mentioned in Section 6.2 was set to 1 by default, and so the system only reported answers to the input questions. By decreasing the parameter, the user can get the system to report tasks generated by itself. In this example, the parameter is set to 0.35.

After being told that "A fruit is a plant", the system is asked "Is a pear a plant?". At this point, the system knows nothing about "pear", and so it produces two derived questions to check whether a "pear" is a "fruit", and whether "pear" and "fruit" are similar concepts (backward inference, see Section 4.4).

Thanks to the low silence parameter, the two questions are reported to the user (and the system is working on them, too). This mechanism makes "active (system-initiated) knowledge acquisition" possible, and now the system and the user use the formal language (defined in Table 3.2) in a symmetric way: both of them can provide knowledge and can ask questions.

In the current example, the user provides the required knowledge indirectly by telling the system that both an "apple" and an "orange" are "fruits", and that "apple" and "pear" are quite similar. Using relevant knowledge ("orange" is irrelevant here), the system gets an answer to the original question by analogy (from "A pear and an apple are similar" and "An apple is a fruit" to "A pear is a fruit") and deduction (from "A pear is a fruit" and "A fruit is a plant" to "A pear is a plant").

Once again we see that in NARS, various types of inference rules are used in a unified manner. An analogical conclusion can later serve as a premise for deduction, or be used to revise an inductive conclusion. As a result, an answer provided by the system is usually generated through the cooperation of various types of rules along its derivation path. Few results are obtained by deduction (or induction, analogy, and so on) *alone*. As was stated earlier, neither the designer nor the system decides in advance what rules to use for a specific task. Without following the system's activity step by step, we even cannot tell how an answer was derived.

- **Example 5**

```
>>> ( 0.7 , 0.6 ) cat (= animal < 1 , 0.8 >

>>> ( 0.7 , 0.6 ) dog (= animal < 1 , 0.8 >

  * 10

>>> ( 0.8 , 0.4 ) cat (= dog < 0.5 , 0 >

  * 8

  <<< ( 0.74 , 0.11 ) cat (= dog < 1.00 , 0.39 >

  * 1

  <<< ( 0.74 , 0.11 ) cat (= dog < 1.00 , 0.39 >

  * 1

>>> ( 0.7 , 0.6 ) dog (= pet < 1 , 0.8 >

>>> ( 0.7 , 0.6 ) dog (= barker < 1 , 0.8 >

  * 50

>>> ( 0.8 , 0.4 ) cat (= barker < 0.5 , 0 >

  * 6

  <<< ( 0.10 , 0.01 ) cat (= barker < 1.00 , 0.12 >

  * 13

  <<< ( 0.22 , 0.02 ) cat (= barker < 1.00 , 0.31 >

  * 11

>>> ( 0.8 , 0.4 ) cat (= pet < 0.5 , 0 >

  * 22

  <<< ( 1.00 , 0.02 ) cat (= pet < 1.00 , 0.31 >

  * 28

>>> ( 0.8 , 0.6 ) animal (= thing < 1 , 0.8 >

  * 17
  <<< ( 0.22 , 0.02 ) cat (= pet < 1.00 , 0.31 >
  * 3
>>> ( 0.9 , 0.6 ) pizza (= thing < 1 , 0.8 >
  * 50
>>> ( 1 , 0.9 ) pizza (= barker < 0.5 , 0 >
  * 434
  <<< ( 0.65 , 0.05 ) cat (= barker < 1.00 , 0.31 >
  * 55
  <<< ( 0.39 , 0.00 ) pizza (= barker < 1.00 , 0.09 >
  * 35
  <<< ( 1.00 , 0.00 ) pizza (= barker < 1.00 , 0.27 >
```

At the outset, the system is told that both “cat” and “dog” are instances of “animal”. Based on this information, the system concludes that “cat” inherits properties from “dog” (“Cat has some dogness”), though the confidence of the conclusion is low. Once again, the evidence is the shared *intension*, not *extension*, of “cat” and “dog”, and the conclusion is produced by the abduction rule.[2]

Later, therefore, given “A dog is a pet” and “Dogs bark”, the system deduces “A cat is a pet” and “Cats bark”, with an even lower confidence value. Though we know the former is true whereas the latter is false, both are equally valid conclusions given the evidence available to the system. When the system obtains more evidence in the future, these beliefs will be revised.

For the same reason, when the system is told that “An animal is a thing” and “Pizza is a thing”, it guesses that “Pizza barks”, because as far as it knows, “Pizza

[2] Of course, the rule also produces the symmetric conclusion “Dog has some catness”, but this is not reported because the symmetric question was not asked, and because the system’s silence parameter is set to 1 by default.

has some dogness", both being "things". Such a conclusion (with its low confidence, of course) is valid, given the system's current knowledge.

Why does this conclusion look ridiculous to us? There are several factors that contribute to such a feeling.

First, we humans know much more about "pizza" and "dog", especially about their differences, than NARS does. If NARS were somehow provided with as much knowledge as we have, it would of course assign a very low frequency value to "Pizza has some dogness", and as a result, "pizza" would not inherit properties from "dog".

Another way of seeing this is to realize that to the system, "pizza" means no more and no less than what it *knows* about the term, and since it knows nothing about cheese and ovens and Italy and eating, the conclusion it makes is not so ridiculous. If the word "seal" had been typed in instead of "pizza", the system would have reached the same tentative conclusion — namely, that "Seals bark" — and in this case it would not have sounded nearly as ridiculous to us (unless we interpret "seal" as meaning "wax seal", in which case it sounds ridiculous again). One has simply to keep in mind that bringing in full human knowledge to interpret the reasonableness of the system's conclusion is very misleading; one has to imagine, instead, that one knows as little about each word, even bland words like "thing", as the system does. This is hard for a human to do, but it is the only way to relate properly to NARS (and in fact, to AI systems in general, if one does not want to succumb to the Eliza effect).

Second, both "A dog is an animal" and "A dog is a thing" are true — we can hardly find negative evidence for either of the two judgments, so their frequency values are both close to 1. However, their confidence values are different. From an extensional point of view, all instances of "dog" will also be instances of both "animal" and "thing", therefore the two judgments are equally supported. From an intensional point of view, however, there are many more properties shared by "dog" and "animal" than by "dog" and "thing". Consequently, "A dog is an animal" obtains more evidence intensionally, and thus has a higher confidence value than "A dog is a thing". This result explains the well-known "specificity principle" — when a concept inherits properties from its superordinate concepts, the more specific a superordinate concept is, the higher priority it has (other things being equal) (Kyburg, 1983; Wang, 1995b).

Third, we humans seldom consider either a dog or a pizza as a "thing" — while they surely *are* "things", this fact is not helpful for most situations. Since *everything* is a "thing", we hardly have anything to say about a "thing". Therefore, it is unlikely for a judgment like "A dog is a thing" to be boosted in importance as a result of its contributions in the past. It follows that the importance value of this piece of knowledge will always be relatively low, and probably will constantly diminish, and thus "thing" will intend to be ignored when the term "dog" is used.

The above analysis shows that though it is logically quite valid for NARS to believe "Pizza barks" in certain situations, such a conclusion will be thrown away as the system acquires more and more experience, bring it closer and closer to that of an English speaker (to whom pizza does not bark). On the other hand, we should realize that, when probing in unfamiliar environments, we humans also often draw conclusions that are similar to "Pizza barks" in spirit — they are valid given our knowledge at the moment, but will look ridiculous later, when more evidence is collected.

The above examples are not the most complex that NARS can handle. More complex experiments, some of them consisting of about twenty terms, a couple of hundred input tasks, and tens of thousand of inference steps, have been carried out in NARS. The results of such experiments provide richer phenomena, but are not as easily analyzable as the above examples.

6.4 Performance

The current implementation of NARS consists of roughly 5,000 lines (in C++), and the object code takes about one megabyte of memory. When it is running on a Sun Sparcstation and has a memory of 25 chunks, each inference step needs about 1/50 second.

The implementation of the logical part is straightforward, so the real purpose of the computer system is to develop and test the control part of the NARS model. Generally speaking, the model works as expected. The resulting system is adaptive to its environment. It is limited by its fixed memory size, can react to various time constraints, and is open to accepting and working on new tasks all the time. By observing the changing internal structure of the memory, we can see how the time–space resources are distributed among competing items, and how the distribution varies dynamically.

Given the complexity and the speed of the system, it is practically impossible for a human user to predict the system's reaction to an interactively provided task. For a simple problem, the user may be able to distinguish possible answers from impossible ones, but is unlikely to be able to predict which possible answers will actually be produced by the system. On the other hand, if a certain state is saved (by the "save" command) and a section of experience thereafter is recorded (by the "record" command), the user can precisely reproduce and study the system's behavior in this period (by resetting the system to the saved state, then feeding the recorded experience to the system).

As an adaptive system, NARS exhibits behavior that is sensitive to its experience. Not only the *contents* of input knowledge matters, the *timing* of input also makes a difference. Two pieces of input knowledge will interfere with each other in terms of the processing that they trigger, if the interval between their arrivals is too short; on the other hand, the chance of their interaction decreases if their arrivals are temporally far from each other.

The input questions, though containing no *knowledge*, provide important *information* to the system by showing which particular inheritance relations are of interest to the environment. The system uses this type of information to adjust the priority and durability values of relevant pieces of knowledge. Therefore, a way to "teach" the system is to ask the right question at the right time. Moreover, repeating a question can also be useful at certain moments, since doing so will draw the system's attention to the question and the concepts involved in it.

In this way, a user of NARS plays a role that is halfway between those of traditional "programmer" and "user". What a user does, when communicating with NARS, influences the system's internal structure and future behavior, but a user cannot *force* the system to believe something or to do something, unless the user has complete control over every bit of experience of the system.

In the future, when a system like NARS is really applied in a domain for some practical purpose, a "training" stage will be necessary, in which a "tutor" (human or computer) provides relevant knowledge to the system, and guides the system to organize it properly by asking proper questions at proper time. For this purpose, a "computer education theory" will be needed, and we can expect it to share many principles with the current education theory (for human beings).

However, NARS 3.0 has been implemented not for some practical purpose, but as a prototype of the theory of intelligence developed in the previous chapters. The implementation can be considered successful in that it was designed according to the theory and exhibits behavior predicted by the theory. Unlike some other AI projects, NARS does not attempt to reproduce psychological data or to solve problems that usually need special expertise.

At its current stage of development, NARS 3.0 has been tested only on small examples involving fewer than 20 concepts. This naturally leads to questions about its potential for “scaling-up”. It is well known that many AI models work well on small problems, yet not at all on big problems. Will this happen to NARS? Obviously, with more tasks and more pieces of knowledge, it would take NARS more time to pick a desired one to work on, but the time for an inference step would be about the same, as was pointed out previously. On the scaling-up issue, there is a basic difference between NARS and many AI projects. When a system is designed under the assumption that an answer may in principle require consulting all relevant knowledge, then the scaling-up problem becomes inevitable — the system can afford the expenses when the knowledge base is small, but cannot do so when the knowledge base is large. By contrast, NARS has been expressly designed under the assumption of insufficient resources, and so it does not attempt to search its knowledge base exhaustively, even when the knowledge base is small. (In fact, if the purpose of the NARS project were merely to answer the questions in the previous examples, a system that simply counted the evidence would work better.) Because NARS takes no advantage of having a small knowledge base, probing the system’s behavior with small examples can still provide us with general insight into the philosophical theory and formal model according to which the computer system was designed.

7

Theoretical Implications

As was stated in Chapter 2, the ultimate goal of this research is not to design a reasoning system *per se*, but to explore the essence of intelligence. The formal model and the computer system have both been developed for this purpose, because with them our discussions can become clearer and more concrete. In this chapter, we return to the theoretical territory to see how several important issues in artificial intelligence and cognitive science are illuminated by this theory/model/system. These issues are not the only ones addressed by the research, but they are among the most representative ones.

7.1 Symbol-grounding

According to NARS' experience-grounded semantics, the meaning of a term consists of its experienced relations with other terms, and determines how the term will be used by the system in the future. A human observer is of course free to interpret the terms appearing in NARS by relating them to words in a natural language or with human concepts, but that is their meaning *to the interpreter*, and has nothing to do with the system itself. For example, if the term "bird" never appears in the system's experience, it is utterly meaningless to the system (though meaningful to English speakers). However, after the sentence "$bird \subset animal < 1, 0.8 >$" appears in the system's input stream, the term "bird" begins to have meaning to the system, revealed by its inheritance relation with "animal". As the system comes to know more about "bird", its meaning becomes, by definition, richer and more complex. The term "bird" may never mean the same to NARS as to a human being (because we cannot expect a computer system to have human experience), but on the other hand, we cannot say that "bird" is utterly meaningless to the system for this (human-chauvinistic) reason.

This leads us to Searle's "Chinese room" argument (Searle, 1980) and Harnad's "symbol grounding" problem (Harnad, 1990). As was mentioned previously, Searle's argument is based on the assumption that a symbol can get meaning only from a model, by an interpretation. If one accepts the idea of an experience-grounded semantics, this is an untenable argument. As soon as a term has experienced relations with other terms, it becomes meaningful to the system, no matter how impoverished or diluted its meaning is. An adaptive system like NARS never processes a term solely on the basis of its "shape" without considering its relations with other terms in the system's experience. The "shape" of a term may be more or less arbitrary, but its experienced relations with other terms are not.

By saying this, I am not claiming that a word in a natural language gets its meaning *only* from its relations to other words in the language, because *human experience* does not consist of words only. The experience-grounded semantics introduced here can be extended to systems with sensory–motor capacity, where truth values of judgments and meanings of terms are no longer determined solely via linguistic interactions among terms, but also to some extent by "nonverbal" components of experience. However, the principles of experience-grounded semantics remain the same

— that is, the truth values of (declarative or procedural) knowledge are determined by evidential support, and the meanings of items in the system's experience (words, perceptual images, motor sequences, and so on) are determined by their experienced relations with other items. In a system of this sort, the meaning of a word is much more complex than in a system whose experience consists of symbols alone, but that fact does not rule out the simpler case as a possible way for symbols to acquire meanings.

The feeling of meaninglessness in Searle's "Chinese room" comes from his deliberate cutting-off of his experience in Chinese from his sensory–motor experience and his experience represented in his native language. If we put an intelligent computer system into the same situation, there are two possible cases. If the computer system already had profound sensory–motor experience and/or a "native language", it might also consider the Chinese characters to be meaningless, because it could not relate them to its previous experience. However, if the system entered the room with no previous experience, Chinese would become its "native language" — that is, the system would build up meanings for the characters on the basis of how they are related to one another, and would not attempt to ground them on some "more fundamental" stuff, nor would it complain about "meaningless squiggles and squoggles" when it failed in doing so. If the system also had sensory–motor capacities and communicated with other similar computer systems in Chinese, we might find that the meanings of Chinese words, to these systems, were as rich and as complex as they are to human Chinese speakers, though it is possible that they might occasionally have differences of opinion about the "correct" meaning of a given word.

If experience-grounded semantics is applied to a symbolic system, all the symbols that the system has are *already* grounded — in the system's experience, of course. The crucial point here is that for a symbol to be meaningful (or grounded), it must somehow relate to the system's environment. However, such a connection need not necessarily be provided by sensory–motor mechanisms. The experience of a system can be purely *symbolic*, or *verbal*, as in the case of NARS. This type of experience is, to be sure, much simpler and more "coarse-grained" than sensory–motor experience, but it is nevertheless experience (Piaget, 1960), and therefore it can serve to ground the symbols that constitute it, just as words in natural language are grounded in human experience.

This suggestion might sound like what Harnad calls "dictionary-go-round" — a disturbing-seeming vision that propels him to express the hope that the meanings of symbols can "be grounded in something other than just more meaningless symbols" (Harnad, 1990). Of course, neither NARS nor a person can learn Chinese from nothing but a Chinese–Chinese dictionary or a Chinese-only radio broadcast. In NARS' current "native language" described in Chapter 3, the "meaning" of the inheritance relations and truth values is hard-wired into the system, and is reflected in the way the relations and values are used by the system. Only the meanings of *terms* are acquired from experience. Thus NARS is not an absolute *tabula rasa*.

There is a subtle difference between Harnad's dictionary-go-round image and the acquisition of symbols' meanings by NARS: in experience-grounded semantics, the meaning of a term is not *reduced to the meaning of other terms* (which would obviously lead to circular definitions in any finite language), but is *defined by its relations with other terms*. These relations are formed during the interaction between the system and its environment, and are not arbitrary at all. Moreover, extending the system's experience to include sensory–motor activities does not fundamentally change this situation. Sensory–motor primitives are still components of a system's experience, rather than components of the "outside world". The meanings that they have (to the system) come totally from their mutual relationships, rather than from their intrinsic physical nature. In many situations, what a perceptual image means to a person depends on its associations with (verbal) concepts.

Human beings judge the truth value of a sentence in the context of their personal experience, and they determine the meaning of a word on the basis of its relations with other words. This is not a new idea to psychologists and linguists (Ellis, 1993; Lakoff, 1988; Palmer, 1981; Tversky and Kahneman, 1974). However, few people have tried to apply this idea to an artificial language defined by a formal grammar. This oversight is due to several assumptions, which, though seldom mentioned, are tacitly accepted by many people.

It is very often implicitly assumed that the semantics of a formal language has to be model-theoretic. Such an inductive conclusion seems warranted by our experience — almost all formal languages have traditionally been assigned their semantics in this way. As a result, people who do not like the semantics usually abandon the language at the same time. However, a language can be "formal" in two different senses. In a *weak* sense, "formal" means merely that the language is artificial, and is defined by a formal grammar; in a *strong* sense, "formal" means that the language is used in conjunction with a model-theoretic semantics. The language used in NARS is "formal" in the weak sense only. From a technical point of view, it would be easy to give the language a model-theoretic semantics, by interpreting the three inheritance relations as in set theory, and interpreting the truth value extensionally only. However, with such a semantics, the language would no longer be suitable for our current purposes.

Logicians, in distinguishing themselves from other scholars, such as psychologists, tend to stress the *normative* nature of logical theory. As a result, in their study of semantics, the goal is often that of looking for *the* real, objective meanings of terms or truth values of sentences. Even if such an opinion has some degree of justifiability when one's purpose is to study the logic of mathematics, that justifiability goes away when one turns to the study of the "logic" of empirical science and common sense. For the purposes of AI, what we need is another kind of normative model, in which meanings and truth values are founded on the system's experience.

7.2 Induction

Induction has been a perennial headache in traditional logic. As Hume's "induction problem" revealed, our predictions of the future cannot be infallible (Hume, 1748). From past experience, no matter how great, we cannot get a perfect description of the "state of affairs", nor can we even know how far our current knowledge is from such an "objective" description. Based on this, Popper drew the well-known conclusion that an inductive logic is impossible (Popper, 1959). However, from our previous discussion of semantics, we can see that what is really being pointed out by Hume and Popper is the impossibility of an inductive logic *built as an axiomatic system*.

Let us see how induction is justified in NARS by considering the following ideal experience: the system gets the knowledge that "$swan \subset bird\ <1, 1>$" ("Swans are birds") and "$swan \subset white\text{-}thing\ <1, 1>$" ("Swans are white"). According to previous discussions, here we find a common instance of "bird" and "white-thing". This experience can be generalized as "$bird \subset white\text{-}thing\ <1, 1/2>$" ("Birds are white") — that is, the inductive conclusion is supported by positive evidence with unit weight. The frequency of the conclusion, 1, means that all evidence summarized by this sentence is positive; the confidence, 1/2, indicates the amount of evidence collected ($c = w/(w + k) = 1/2$, with $w = 1$ and $k = 1$). The truth value of the conclusion does not measure how many birds "in the world" are white. If in another section of experience the system knows that crows are birds but that they are not white, it similarly derives the inductive conclusion, "*bird* $\subset$ *white-thing* $<0, 1/2>$", from that piece of evidence alone. When these two conflicting conclusions meet, their underlying bodies of evidence are combined by the system's revision rule, and the system produces a summarized conclusion "*bird* $\subset$ *white-thing* $<1/2, 2/3>$", where the frequency is a weighted sum of the competing two, and the confidence is higher than the premises', due to the accumulation of evidence.

From the above example we can see that the inference rules of NARS are not truth-preserving in the traditional sense, since the conclusions may conflict with new evidence; however, they are truth-preserving according to the new definition of truth value, because the truth value of the conclusion is determined by the experience summarized in the premises.

If the answers provided by NARS are fallible, then in what sense are these answers "better" than arbitrary guesses? Considerations of this sort lead us to the concept of "rationality". When infallible predictions cannot be obtained (due to insufficient knowledge and resources), answers based on past experience are nonetheless better than arbitrary guesses, provided the environment is *relatively stable*. If the environment is completely instable, any attempt to adapt to it will be hopeless. The fact that an answer is only a summary of past experience (thus no future confirmation can be guaranteed) does not make it tantamount to an arbitrary conclusion — indeed, that is what "adaptation" means.

Someone might argue that what NARS does is not genuine "induction", which (by definition) is "to generate a *universal* statement from specific statements", while what NARS gets is "statistical statements". Though such a definition of induction is widely accepted, and many discussions are based on it (Popper, 1959), it is not an appropriate way to study induction. From a theoretical point of view, such a definition presupposes the use of a formal language, such as first-order predicate language, where every general statement is created by using a "universal quantifier". This is a particular way to formalize Hume's problem, but by no means the only reasonable one. What Hume was concerned with is how we can extend our finite specific past experience to the infinite general future situation. And yet, from a practical point of view, almost none of our knowledge comes with a "universal quantifier" attached, except (perhaps) in mathematics. When we say "All ravens are black", what we mean is "All ravens (according to our knowledge) are black", but not "All ravens (whether known or unknown to us) are black". For a system with insufficient knowledge (either a human or a computer), the latter statement cannot be justified. If we insist upon restricting the term "induction" to this limited meaning, we have to admit that it is an unattainable goal. However, by doing this, we prevent ourselves from studying the real problem — that is, how to use past experience to predict future experience, while being fully aware that the predictions will not be infallible.

The study of induction in artificial intelligence has been strongly influenced by the Popper tradition. Representative works in the field define "induction" as the process by which a general statement (usually a universally quantified sentence in a first-order language) is found. From it, specific cases can be deduced (by first-order predicate logic) (Michalski, 1983; Quinlan, 1986). Therefore, induction is often referred to as "reverse deduction". What distinguishes NARS from these approaches to induction is the following list of properties:

1. In NARS, general judgments do not necessarily have binary truth values, but can have counterexamples. Knowing that penguins are birds but cannot fly does not falsify the judgment "Birds fly", but only *decreases its frequency*, thus making the judgment "weaker".

2. In NARS, induction is not carried out by an algorithm that generates a hypothesis by searching a "space of possible hypotheses", then exhaustively uses all

available knowledge to test it. Instead, as is shown by the previous examples, each piece of evidence generates an inductive conclusion, and assigns a truth value to it at the same time.

3. In NARS, inductive conclusions are generated and revised incrementally, so they can be produced under various time pressures.

4. In NARS, induction is not an isolated “tool” or “module” that can be invoked by the user to solve a domain problem, but an operation intimately related to and inseparable from other inference rules. As is shown by the examples, inductive conclusions merge by the revision rule, and the premises and conclusions of induction are provided by and are used by deduction, abduction, analogy, and so on. Indeed, a major reason for my having chosen term logic over predicate logic is that in term logic, multiple types of inference can be naturally integrated with one another.

7.3 Categorization

Among other things, NARS is a model of categorization, in which a term can be understood as the name of a concept or category (Wang, 1993b).

As was explained previously, the basic relationships between two terms are the three inheritance relations, which in different ways indicate the extent to which one term can be used as the other, or, in Hofstadter’s terminology, “slipped” into the other. In this sense, NARS attempts to formalize the “as” or “slippability” relations between concepts, and to give them a central place in the study of artificial intelligence (Hofstadter, 1995).

In NARS, inheritance is always a matter of degree, determined by the system’s experience with the terms in the relation. In this aspect, it agrees with fuzzy logic (Zadeh, 1965) and several psychological models of categorization (Medin and Ross, 1992). The meaning of a given concept — that is, what makes it different from all others — is its unique position in the system's experience. This position is revealed by its relations with other concepts — that is, by its extension and intension, defined in Chapter 3. This feature makes NARS different from the classical theory of categorization, according to which concepts have defining features that act as criteria for determining category membership, and the "membership" relation between a concept and potential exemplars is binary (i.e., black-and-white).

Since all inferences in NARS involve inheritance relations, we can say that all the activity in the system amounts to categorization or high-level perception. When the system is looking for an answer to a question "$S \in P$", what it does is nothing but trying to judge the degree to which S is an instance of P, or identically, the degree to which P is a property of S.

How is such a judgment made? In NARS, it depends on the current meaning of the two concepts. If both S and P are characterized by a set of properties, the degree of membership can be judged by *abduction* — that is, by looking at the extent to which S has P's properties. If P is largely characterized by a set of exemplars, then the degree of membership can be judged by *analogy* — that is, by looking at the extent to which S is similar to P's most typical, or strongest, instances. The degree of membership can also be judged by *deduction* if there is a term M such that the truth values of both "$S \in M$" and "$M \subset P$" are known. Usually, in fact, the degree of membership is judged in more than one way, and the evidence from all the different sources is combined (by the revision rule) to get a summarized conclusion.

This NARS-oriented account of categorization can be characterized as a generalization of current psychological theories on the issue, of which "prototype theory" and “exemplar theory” are two prototypical exemplars. We discuss them both very briefly.

According to *prototype* theory, a concept is characterized by properties shared by most of its members. As a result, category boundaries become fuzzy in the sense that some exemplars, thanks to the fact that they share more properties with the “prototypes” of the concept, are more “typical” members than others. The prototypes themselves are determined by the central tendency of the concept’s members (Rosch, 1973).

According to *exemplar* theory, a concept is determined by a set of examples, and the degree of membership of a test item in the concept is measured by its similarity to particular examples that exemplify the concept. In this theory, membership is fuzzy, too, but for different reasons than in prototype theory. Here, the fuzziness comes from the fact that the similarity relation usually is a matter of degree (Nosofsky, 1991).

Both theories represent special cases of categorization in NARS. By considering both the extension and intension of a concept, and by using multiple types of inferences, NARS provides a more general model of categorization.

Another important nature of NARS is its assumption of insufficient resources. In the context of categorization, this means that when the strength of a membership relation is assessed, only part of the relevant knowledge can be used. But since we want to give all possibilities a chance, and at the same time to focus on what has proven most useful in the past, a probabilistic mechanism is used for selection among relevant but competing pieces of knowledge, as described in Chapter 5. Moreover, the probability distribution is continually adjusted as the situation changes.

Taken all together, these ideas give each concept in NARS a context-sensitive dynamic structure. As the system runs, new relations are added, old relations are deleted, and the priority distribution is adjusted by the system to adapt to its environment. As a result, when a concept is used at different times, its meaning can be quite different.

The internal structure of a concept in NARS is dynamic but not arbitrary. When the system has plenty of experience with a particular concept, some of its relations become stable (i.e., have a high durability value), and have a high probability of being addressed when the system uses the concept. Let us call these the *dominant relations*, or *essence*, of the concept. They endow the concept with a kind of "hard core", though in fact even the core slowly changes over time.

According to this view of categorization, the meaning of a term is in principle "personalized" or "idiosyncratic" — that is, it is different in different systems, despite bearing the same name. The "objective" aspect, or more accurately, the "public" aspect, of a term's meaning is a natural outgrowth of communication. Systems that coexist in the same environment will exert influences on each other concerning what each term "means". For a system to communicate efficiently, it has to try to take into account how a term is used by the other systems. Here we thus see two contradictory tendencies: individual experience tends to make the meanings of terms even more personalized and *unique*, while communication tends to reduce these interpersonal differences and to make meanings ever more *universal*.

As a result, concepts in NARS approach the idea that Hofstadter calls "fluid concept" — that is, "concepts with flexible boundaries, concepts whose behavior adapts to unanticipated circumstances, concepts that will bend and stretch — but not without a limit" (Hofstadter and FARG, 1995). The "Slipnet" structure, developed by his group, also represents concepts by their relations between each other, and allows the relations to change dynamically (Hofstadter and FARG, 1995). Consequently, concepts in the Slipnet are fluid while still having relatively stable "hard cores", similar to concepts in NARS.

7.4 Is NARS a logic?

As we have seen, concepts in NARS are usually fluid, fuzzy, and elusive; judgments are usually equivocal, conflicting, and fallible; and the system's behaviors are usually unpredictable and irreproducible. The system may be absent-minded, may forget important information, and may change its mind from time to time. How can we still say that NARS works according to a *logic*?

In fact, the concept "logic" has two different senses (Wang, 1992). Construed very broadly, logic is the set of principles of, and criteria for, valid inference. However, nowadays the concept is used in AI under a narrower construal, which is restricted to first-order predicate logic, its variations, model-theoretic semantics, theorem-proving, and so on (Birnbaum, 1991; McDermott, 1987; Nilsson, 1991).

Obviously, NARS does not constitute a logic in this narrower sense. However, it is a logic in the broader sense. Technically, it has a formal language, and uses a set of formal rules to do inference. Theoretically, it is an attempt to formally capture the principles of valid inference. From Chapter 2, we know that NARS is still a *normative* model of reasoning, rather than a *descriptive* model. That is, NARS represents what a system (human or computer) *should* do, in a given situation and following definite principles of reasoning. However, since the environment assumed by the model is similar to the circumstances in which the human mind finds itself, we can expect NARS to be relatively close to a descriptive model of human reasoning, at least when compared with other normative models, such as Aristotle's logic, first-order predicate logic, and the Bayesian school of probability theory (Wang, 1996a).

To clarify more sharply the difference between NARS and other reasoning systems, we now define three categories of reasoning systems, based upon their assumptions about the sufficiency of knowledge and resources:

Pure-axiomatic systems. Such systems are designed under the assumption that both knowledge and resources are sufficient (with respect to the questions that will/can be asked), so adaptation is not necessary. A typical example is the notion of "formal system" suggested by Hilbert (and many others), in which all answers are deduced from a set of axioms by a deterministic algorithm, and which is applied to some domain using model-theoretical semantics. Such a system is built on the idea of sufficient knowledge and resources, because all relevant knowledge is assumed to be fully embedded in the axioms, and because questions have no time constraints, as long as they are answered in finite time. If a question requires information beyond the scope of the axioms, it is not the system's fault but the questioner's, so no attempt is made to allow the system to improve its capacities and to adapt to its environment.

Semi-axiomatic systems. Such systems 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. For example, non-monotonic logics draw tentative conclusions (such as "Tweety can fly") from defaults (such as "Birds normally can fly") and available knowledge (such as "Tweety is a bird"), and revise such conclusions when new knowledge (such as "Tweety is a penguin") arrives. However, in these systems, defaults and pieces of knowledge are usually unchangeable, and time pressure is not taken into account (Reiter, 1987). Many learning systems attempt to improve their behavior, but still work solely with binary logic where everything is black-and-white, and persist in always seeking *optimal* solutions of problems (Michalski, 1993). Although some heuristic-search systems look for less-than-optimal solutions when working within time limits, they usually do not attempt to learn from experience, and do not consider possible variations of time pressure.

Non-axiomatic systems. In this kind of system, the insufficiency of knowledge and resources is built in as the ground floor, and the system works accordingly.

According to the working definition of intelligence proposed in Chapter 2, pure-axiomatic systems are not intelligent at all, non-axiomatic systems are intelligent, and semi-axiomatic systems are intelligent in certain respects.

In some practical situations, an intelligent system is not necessarily superior to a nonintelligent system. In fact, quite to the contrary: when a problem can be solved by both types of system, the nonintelligent method is usually better, because it guarantees a correct solution. As Hofstadter once observed, for tasks like adding two numbers, a "reliable but mindless" system is better than an "intelligent but fallible" system (Hofstadter, 1979).

Pure-axiomatic systems are very useful in mathematics, where the aim of study 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 manner so accurate and reliable that the procedure can be reproduced by an algorithm. We need intelligence only when no such pure-axiomatic method can be used, due to the insufficiency of knowledge and resources. For similar reasons, 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.

Logic was originally the study of human reasoning in general situations. However, since the rise of "mathematical logic", the attention of logicians has been focused on the foundations of mathematics, an area that involves different principles from those that underlie everyday thinking (Ellis, 1993). Logicians' studies have contributed greatly to mathematics, but have left a misleading heritage to AI. When AI researchers eagerly adopt standard tools from mathematical logic, what is too often ignored is the fact that these tools were originally conceived and built for completely different purposes.

Traditionally, AI has been referred to as a branch of computer science. According to our previous definitions, AI can be implemented with tools provided by computer science, but from a *theoretical* point of view, AI and computer science make opposite assumptions: computer science focuses on pure-axiomatic systems, but AI focuses — or *should* focus — 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 the theory of Turing machines). These theories take sufficient knowledge and resources for granted, and therefore adaptation, plausible inference, and tentative solutions to problems are neither necessary nor possible.

Similar assumptions are often made by AI researchers with roughly the following type of justification: "We know that the human mind works under conditions of insufficient knowledge and resources, but if you want to set up a *formal* model and then implement it as a *computer* system, you must somehow *idealize* the situation."

It is true that every formal model is an idealization, and so is NARS. The key question concerns 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 such insufficiency is a *defining* feature of intelligence, so if it were abandoned in the "idealization", the resulting study, whatever its worth might be, would be about something other than intelligence.

Some critics implicitly assume that because a certain level of a computer system can be captured by first-order predicate logic and implemented as a Turing machine, these axiomatic theories also apply to the entire range of behavior that such a system can exhibit (Dreyfus, 1992; Penrose, 1994). This is not the case. When a virtual machine A is implemented on a virtual machine B, the former does not necessarily inherit all the properties of the latter. For example, it is invalid to conclude that a computer cannot process decimal numbers (because the hardware uses binary numbers), cannot process keyboard characters (because underneath it all, everything is just bits), or cannot use a functional or a logical programming language (because the commands of such languages are eventually translated into procedural machine language).

Obviously, with its fluid concepts, revisable knowledge, and fallible inference rules, NARS violates all the norms of classical logics. However, as a virtual machine, NARS can be built upon another virtual machine that is in fact a pure-axiomatic system, as my implementation demonstrates, but this fact does not in any sense make NARS "axiomatic".

Many arguments proposed against logical AI (Birnbaum, 1991; McDermott, 1987), symbolic AI (Dreyfus, 1992), or AI as a whole (Searle, 1980; Penrose, 1994), are actually arguments against a more restricted target: pure-axiomatic systems. These arguments are powerful in that they reveal many aspects of intelligence that cannot be produced by a pure-axiomatic system (though these authors do not use this term), but some of the arguments seriously mislead by portraying such systems as the prototype of AI models of mind.

7.5 NARS as an inheritance network

By working on a reasoning system with its formal language and inference rules, one does not necessarily commit oneself to the assumptions of traditional logic-based AI paradigms. Designed as a *reasoning system*, but not a "logicist" one (McCarthy, 1988; Nilsson, 1991), NARS actually shares more philosophical assumptions with the *subsymbolic* or *connectionist* movement (Hofstadter, 1985; Holland, 1986; Holland et al., 1986; Rumelhart and McClelland, 1986; Smolensky, 1988), despite the fact that I chose to formalize and implement these assumptions in a framework that on the surface looks closer to the traditional symbolic-AI tradition.

In fact, NARS can be naturally described as an *inheritance network* (Wang, 1994c). We can see each term as a *node*, each judgment as a *link* between two particular nodes, and the corresponding truth value as the *strength* of the link. Between two given nodes, there are three possible types of links, corresponding to the three inheritance relations — namely, "$\longrightarrow$" for "$\subset$", "$\longleftrightarrow$" for "$=$", and "$\longmapsto$" for "$\in$". In this way, the knowledge base of NARS is a network like Figure 7.1. This type of network is different from a semantic network and other symbolic networks, because there are only three types of link, whose meanings and functions are precisely defined (as was done in Chapter 3), and because there are often two or more links between a given pair of nodes, and their truth values might conflict with each other.

The network has both *active links* (tasks) and *passive links* (knowledge). Priorities are defined among nodes, active links, and passive links. In each atomic step of processing, an active link interacts with an adjacent passive link to generate new links, and different types of inference correspond to different combinations of the two component links (Minsky, 1985; Wang, 1994c), as summarized in Figure 7.2, where the solid links are premises, and the dashed links are conclusions. The corresponding truth-value functions are listed in Table 4.3.

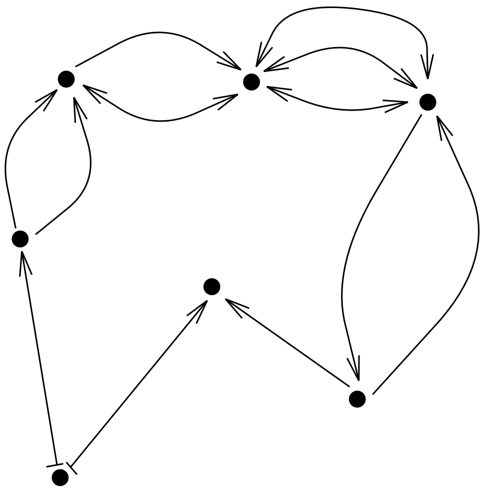


Figure 7.1: An inheritance network.

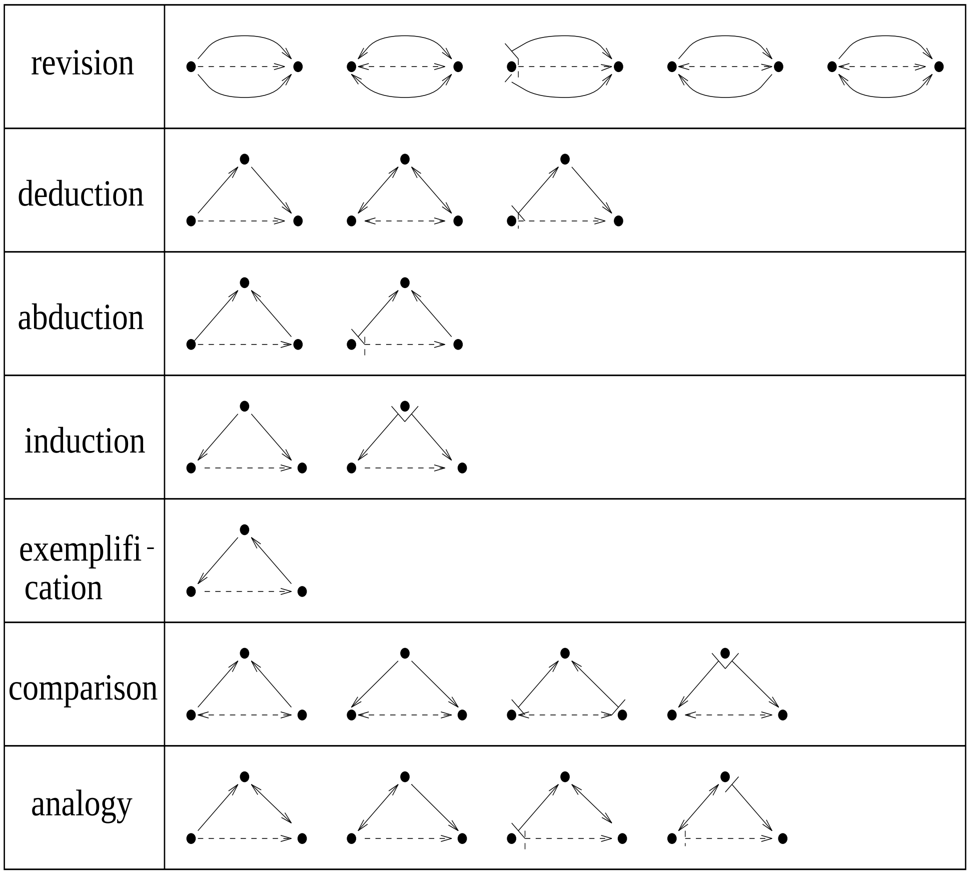


Figure 7.2: Rules of the inheritance network.

In such a network, to answer a question means to determine the strength of a link, given its type and its beginning and ending nodes, or else to locate that node that has the strongest link the system can find from or to a specified node. Thanks to the fact that during the processing, not only the topological structure of the network but also the strengths of its links and its priority distribution are all constantly changing, what the system does is much more than searching a static network for the desired link or node.

Seen as a network, NARS shares many properties with various subsymbolic approaches (Hofstadter and FARG, 1995; Holland, 1986; Smolensky, 1988), such as parallel processing, nondeterminism, self-organization, distributed representations, and so on. Let us consider the last property in more detail here.

At first glance, the internal representations of NARS seem to be local, since the knowledge "Swans are birds" is stored explicitly in the link that runs between the node $swan$ and the node $bird$. However, when the system is told "Swans are birds", the judgment is treated both as a passive link (to be set up between the given nodes) and as an active link (to interact actively with other links); therefore, it will have effects on other nodes and links. On the other hand, when the system is asked "Are swans birds?", it will not only seek the direct link between nodes $swan$ and $bird$, but will also try to answer the question by inference from available knowledge, so that other nodes and links may be involved in the process.

As a result, NARS' internal representations become *distributed* in the following senses: (1) an input task may have non-local effects, (2) an output result may have non-local sources, and (3) local losses of information (whether by intentional deleting of information or by unintended hardware malfunction) may be partially recovered from information kept in other parts of the system.

Since NARS can be naturally described as a network, why is a "logical" description still preferred? Indeed, the terminology of networks, using concepts such as “node”, “link”, “strength”, “activation”, and so on, is semantics-neutral and can therefore be used for many different purposes without stirring up any controversy, whereas philosophical concepts like “meaning”, “truth”, and “induction” tend to stir up hornet’s nests of argumentation.

So why do I use this “logical” language as my primary way of presenting and describing NARS? By constantly using the terminology of logic, what I want to show is: to model intelligence faithfully, what really matters is the set of *underlying theoretical premises*, not the terminology or the technology. Once one accepts a new working definition of intelligence, one finds that a reasoning system, despite having the trappings of a formal language and truth-preserving inference rules, can still have many interesting and unexpected properties. The basic troubles with traditional logic-based symbolic AI stem from its fundamental assumptions about sufficiency of knowledge and resources, and its pursuit of completeness, consistency, and decidability, rather than from detailed decisions about how AI systems work. It seems to me preferable to consider NARS as a type of logic, due to its preciseness, during the design process and also in explaining the basis for its design, whereas the “network” image seems preferable, due to its vividness, for occasions when the system must be presented in a very short time to other people.

7.6 Is it still computation?

With the control mechanisms described previously, it is easy to see that the processing triggered by a task given to the system is context-sensitive. Even for the same task, with the same priority and durability values, the result may be different (though not necessarily so). How a task is treated depends on what knowledge the system has, how the knowledge is organized, and how much of the system's resources the task gets — put simply, it is determined by the system's experience, which includes not only events that take place before the task showed up, but also events that happened after that moment. The contents, the order, and the timing of events all matter. Furthermore, a question may result in no answer, one answer, or more than one answer.

As a result, the response of NARS to a given task is *unpredictable* from that task *alone*. However, it needs to be stressed that the unpredictability is not caused by the "take-out" operator of the "bag" class, which makes choices probabilistically. That operator is used for distributing resources *unevenly* among competing items, rather than for introducing a "pure random" or "nondeterministic" factor into the system solely for the sake of making it unpredictable. Similar points are made in (Hofstadter, 1993; Hofstadter and FARG, 1995). Indeed, it is perfectly reasonable to implement that operator deterministically, something that can be easily done using standard pseudo-random-number generators. When the system is reset to its initial state, and the previous experience (i.e., input tasks) is precisely reproduced, the system will simply repeat what it did before, including all of its probabilistic choices involving bags.

An interesting and important question about NARS naturally arises: Is the system still *computing*?

According to the usual definition of a Turing machine M, a *computation* is the procedure by which M transforms its initial state q_0 into q_f, one of its potential final states, in response to input data d_i. In its final state, M provides an output d_o as the result of the computation. We can equally well say that M is a *function* that maps d_i to d_o, or that M is an *algorithm* with d_i and d_o as input and output, respectively.

A conventional computing system works in a sequential and deterministic way:

waiting for the user to input a new task;
accepting a task;
processing that task;
reporting the result;
resetting the working memory;
waiting for the user to input a new task;
and so on, cyclically.

According to the definition of *computation*, the characteristics of such a working mode are:

1. There is a unique *initial state* in which the system can accept input tasks, and tasks are processed in a one-by-one manner. If a task arrives when the system is still busy with another task, the new task has to wait. Even if interrupt mechanisms are taken into consideration, the picture is fundamentally the same.

2. The system always yields the same result for a given task, no matter when the task is processed.

3. The amount of resources spent on a task is a function of the task alone, depending on the complexity of the algorithm and the amount of relevant knowledge, but independent of when the task is processed.

4. There is a predetermined set of *final states* in which the system will stop working on a task and provide a result, no matter whether there are other tasks waiting to be processed.

As has been previously made abundantly clear, NARS does not work in this way.

In NARS, there is no unique “initial state” in which the system waits for and accepts new tasks. At any moment when the system is running, tasks can be accepted, in many different internal states.

Similarly, there is no “final state” for a task. For instance, if a task’s urgency is low (relative to other tasks), it is even possible for it to be completely ignored. If a tentative answer to a question is reported, usually neither the system nor its human designer can predict whether a better answer will be reported later, taking more knowledge into consideration. It is undecidable whether a given answer is *the* answer to the question, since that will depend on events still to take place in the future, such as whether the system acquires from the environment new knowledge related to the task, or whether more time winds up being spent on it.

By slightly changing the meaning of the term, one might say that NARS has an initial state — namely, when its memory is completely empty (the system is “born” without any innate domain knowledge). Its state changes as soon as it interacts with its environment and begins processing tasks. The system never will return to its initial state, until and unless a user terminates the processing and erases all of its memory. In such a case, the system can of course be “reborn” with the same “genetic code” — its sets of inference rules, control mechanisms, “personal parameters”, and so on. However, unless the experience of the system perfectly repeats its experience in its “previous life”, the system’s behaviors will be different.

In summary, the system’s behaviors are determined by its initial state and its experience, but not by either one of the two alone.

If we take a *question* (as defined in Chapter 3) as the input to NARS, and the answer to that question as output, then their relation cannot be captured by concepts like “computation”, “function”, and “algorithm”. However, this does not mean that there are “magical” or “pure random” factors introduced in NARS. The system can still be implemented as a Turing machine, and therefore it is still computing, but at a different *level of description*.

If we take an arbitrary state of NARS, q_1, as an “initial state”, the state the system arrives at after a certain amount of time, q_2, as a “final state”, then we can view what NARS does during that period of time as *computation*, with its experience (all of the tasks provided by the environment during that time) as the input, and its responses (all of the system-generated reports) as the output. At this level of description, the system still works as a function or algorithm that deterministically maps an input to an output. However, this view of input and output is very unwieldy and unnatural.

In summary, the behavior of NARS can be described at two different levels. At one of them, NARS is indeed computing, but not at the other. This state of affairs has been articulated by Hofstadter in the following way: “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 modeled on the machines we call ‘computers’ ” (Kugel, 1986). In contrast to this, conventional computer systems, while also describable at these two levels, are computing in both of them. Let us use an ordinary sorting program as an example: you can take either a single sorting problem (the analogue to a single question from the environment), or a *sequence* of such problems (the analogue to the unwieldy and long sequence of user inputs in a given period of time), as the input, and the processes in both cases are computation — the program’s response to a given sorting task is fully determined and does not depend on the processing of other sorting tasks, or the answers given to them.

For practical purposes, what we are interested in is usually the relationship between questions and answers. From the above discussion, we see that in NARS such a relationship cannot be referred to as computation. On the other hand, NARS can still be implemented in abstract Turing machines as well as in concrete von Neumann computers.

This conclusion is not just a picturesque new way to see things, but has important methodological implications. When a system like NARS is designed, the designer should not try to decide what answer the system should produce in response to a given question — that should be decided by the system itself at run time; the designer simply cannot exhaustively consider all possible situations in advance (the designer, hopefully, is also an intelligent system, thus limited by insufficient resources). For similar reasons, the designer cannot decide in advance how much of the resources to spend on a certain task, for this is totally context-dependent. Thus, the designer is no longer working on either domain-specific algorithms or general-purpose algorithms (like GPS), but rather on *meta-algorithms*, which carry out inferences, manage resources (like a small operating system), and so on (as described in the previous chapters). In this way, the problems solved by the designer and the problems solved by the system itself are clearly distinguishable from one another.

Although general-purpose algorithms and meta-algorithms are both independent of specific domains, there is still a fundamental difference between the two types of algorithm, with respect to how a domain problem (i.e., a question asked by the user) is solved. In the former cases, the problem-solving process is still computation. As was described previously, the system accepts the problem at its initial state, processes it according to predetermined procedure, then stop at the final state and reports the solution. We already know that NARS does not work in this way. The fact that NARS still consists of a set of algorithms does not mean that the system's problem-solving activities (for user provided problems) follow any algorithm. Of course, the algorithms in NARS do facilitate the problem-solving activities, but in a different way. For example, Section 5.3 actually describes the algorithm that controls an atomic step, which invoke other algorithms, like the put-in and take-out procedures of various bags. However, none of these algorithms takes user-provided problems as its input. In fact, armed with these algorithms, NARS deals with environment-provided tasks *without* (task-oriented) algorithms. We call them meta-algorithms, because they are not ready-made methods for user problems, but (ready-made) methods by which the "object-level" methods can be formed dynamically in run time.

These ideas allow us to explain why *Tesler's Theorem* — "AI is whatever hasn't been done yet" (Hofstadter, 1979) — applies to many AI projects: in those projects, the designers usually use their own intelligence to solve domain problems, and then implement the solutions in computer systems in the form of task-specific algorithms. The computer systems then execute the algorithms on specific instances of the problems, an activity that can hardly be referred to as "solving problems intelligently". For example, many "expert systems" have no learning ability. Such systems are designed by "knowledge engineers", who abstract domain knowledge from experts in a particular field, and then implant this knowledge into a computer system, so as to reproduce the experts' problem-solving ability. According to my working definition of intelligence, both the domain experts and the knowledge engineers are intelligent — they work with insufficient knowledge and resources, and they learn from their experience — and yet the expert system itself is not intelligent, because, ironically, when it faces a problem, it faithfully follows the predetermined algorithms that were abstracted from the experts' intelligent behaviors, thanks to the intelligence of the knowledge engineers.

This new way of describing AI also changes what we usually referred to as a "solution". Let us take the "combinatorial explosion" problem as an example. If, for a particular problem, there is an algorithm that takes an amount of time that grows exponentially with some parameter in the problem, usually such an algorithm is useless in actual practice — the time expense will rapidly increase to astronomical figures, and the system will simply be paralyzed.

The traditional way to deal with this problem is to look for a faster algorithm, even if that implies sacrificing the quality of the solution. Since in NARS, problem-oriented algorithms are not used, the very concept of "computational complexity" disappears. If the system is faced with a problem that may take a large amount of time, what is guaranteed is not that the system will arrive at a satisfactory solution, but rather, that the system will not be paralyzed by the problem — the system will gradually decrease the problem's priority, while still leaving it a chance to be solved through future inspirations. This is much like what happens in the human mind: we say there is no "combinatorial explosion" in our minds, not because we can solve all problems in polynomial time, but because we seldom, if ever, stick to exhaustive searching, nor even to working monomaniacally on a single problem facing us.

NARS is creative and autonomous in the sense that its behavior is determined not only by its initial design, but also by its "personal" experience. It can generate judgments and questions never anticipated by its designer, and can work on them by its own choice. A "tutor" can "educate" it by manipulating its experience, but cannot completely control its behavior due to the complexity of the system. From a pragmatic point of view, this is neither necessarily a good thing, nor necessarily a bad thing. It is simply the case that an adaptive system with insufficient knowledge and resources has to behave in this way.

8

Conclusion

In this chapter, the major results of the NARS project are summarized, the foundations of the research are evaluated, and its limitations and possible future extensions are discussed.

8.1 Major results

"Intelligence" represents certain information-processing principles, which are required to deal successfully with certain environments.

Concretely, intelligence is adaptation under insufficient knowledge and resources, which means that the system must be finite and open, and must work in real time.

A reasoning system provides a suitable (though not the only possible) avenue for the study of artificial intelligence.

Model-theoretic semantics is not applicable to situations where intelligence, in the above sense, is required. For this reason, an experience-grounded type of semantics is introduced instead, which helps both the designer's building of the system and the user's understanding of the system. In the framework of this semantics, the truth value of a judgment is determined by its evidential support, and the meaning of a term is determined by its experienced relations with other terms.

To support such a semantics, a term-oriented language is chosen, which is characterized by the use of subject–predicate sentences. A judgment, indicating an inheritance relation between some pair of terms, is such a sentence with a truth value attached, which indicates the extent to which the subject belongs to the extension of the predicate, and the predicate belongs to the intension of the subject. Evidence for the sentence, either positive or negative, can be collected both from the extensions and from the intensions of the terms, spread about in the rest of the judgments of the system. As a result, each judgment's truth value, determined by available evidence, represents various different types of uncertainty in a uniform manner.

All inference in NARS is about inheritance relations among terms. NARS' inference rules were constructed by taking both extensions and intensions into account, and by considering all possible types of combinations of premises. As a result, NARS has a set of inference rules for revision, deduction, induction, abduction, exemplification, comparison, analogy, and backward inference. These different types of inference are carried out in a basically uniform format, are justified by the same semantics, and are used in similar ways.

To work in real time and with insufficient resources, the system processes many tasks in parallel. The resource of time is distributed among the tasks unevenly, and the distribution is dynamically adjusted when the situation changes. Tasks that are more relevant to the system's current priorities are always given more processing time, so that in effect, they are processed faster and their implications (in the case of pieces of knowledge) or their potential solutions (in the case of questions) are explored further than happens for less relevant tasks.

The system's memory is organized into a two-level structure: the task/knowledge level and the chunk level. At the chunk level, tasks and pieces of knowledge are clustered according to the terms appearing in them. On each level, items are stored in "bags", which are data structures characterized by fixed size and supporting probabilistic retrieval of items within them. A chunk is a high-level unit of resource allocation and inference activity. Chunks cooperate by sending messages (tasks) to one another.

The system works by repeatedly executing atomic inference steps, each of which takes only a very short time. New questions and knowledge can be accepted at any time, and their processing will depend on the current state of the system's knowledge and resources.

Such a model of intelligence can be implemented on a computer system, using currently available hardware and software technology. The resulting system is not complex technically, but does produce complex behaviors, as predicted by the theory.

This theory/model/system has relevance to many important philosophical issues in cognitive science and artificial intelligence, such as symbol-grounding, induction, categorization, the role of logic and computation in thought, and others. In fact, a noteworthy aspect of this research project is its tight connection with many questions that are traditionally studied in separate disciplines and subdisciplines, and the coherence of its answers to these questions.

8.2 Evaluation

After many descriptions and discussions, let us reevaluate the foundations of the research, by comparing the working definition of intelligence with the requirements set up at the beginning of Chapter 2:

Faithfulness. Obviously, natural information-processing systems (i.e., humans and animals) are adaptive, and they have to work with insufficient knowledge and resources. Being more adaptive, human beings are much more intelligent than other animals. By contrast, though traditional computing systems also have extremely limited knowledge and resources, they are usually set a carefully limited class of problems, chosen so that their knowledge and resources will in fact be *sufficient* for those problems. Therefore, the definition draws a line between intelligent and nonintelligent systems that is faithful to the common usage of the word "intelligence."

Sharpness. The definition is sharp, because whether a system is adaptive can be determined by testing whether its behavior depends on its experience. For a computer system, whether it is designed under the assumption of insufficient knowledge and resources can be determined by checking for three properties: finiteness (can the system forget?), operation in real time (can the system work under a range of different time constraints?), and openness (does the system restrict what it can be told or asked?). As Turing proposed, we still decide whether a system is intelligent by "talking" with it, but the standards are different — we do not require the system to talk like a human.

Fruitfulness. As the foregoing chapters have demonstrated, the definition has yielded fruit by inspiring the major components of NARS, which have exhibited many desired properties. Indeed, rooted in this definition of intelligence, NARS addresses many facets of AI in a consistent manner, and also provides a way of conceiving AI that clearly distinguishes it from related disciplines, such as computer science, psychology, and neuroscience.

Simplicity. The definition is quite simple, making it easy to discuss and to apply to research. Its direct outcome, NARS, is also relatively simple in its structure (compared with other AI systems), though the system's behavior can be very complex due to its interaction with its environment.

Because of these considerations, I believe that the working definition of intelligence introduced in this project is preferable to many others accepted by AI researchers. However, I do not claim that the definition is *the correct one.* Obviously, there are many intelligence-related phenomena that have not been not covered by the current version of NARS. These phenomena suggest extensions of NARS, which may cause future revisions to the definition, but which at this time cannot be used as arguments against the definition. A working definition, as the cornerstone of a research paradigm, merits rejection only when a superior rival is offered, not merely when a weakness is pointed up in it (Kuhn, 1970). I hope that in the near future the working definition of intelligence used here can be replaced by a better one, one that will still be based on the four criteria of *faithfulness, sharpness, fruitfulness*, and *simplicity.*

8.3 Limitations and extensions

We can distinguish three types of limitations for NARS: ones that *can* and *will* be removed in future versions of system, ones that *might* be removed in future versions of the system, and ones that *cannot* be removed in future versions of the system.

- **The next step: NARS 4**

NARS 3.0 is an intelligent reasoning system, in the sense that it is an adaptive system that works under insufficient knowledge and resources. However, this does not mean that it cannot be made "more intelligent". Indeed, according to my working definition of intelligence, one can increase a system's intelligence by extending its interface language, by providing it with new inference rules, by increasing its efficiency at resource management, by giving it the capacity to self-organize at a higher level, and so on.

The most obvious limitation of NARS 3.0 is in its language and inference rules, where only simple (atomic) terms can be represented and processed. In the next version, NARS 4, *compound terms* will be introduced into the system. Specifically, these types of compound terms will be defined and integrated into the processing of NARS.

1. If S is a term, $\{S\}$, the set containing S, is also a term. Actually, this compound term was already introduced in Section 3.5, when the "$\in$" relation was defined. However, the corresponding compound terms are not directly processed in NARS 3.0.

2. If S and T are terms, $(S \cup T)$ (*union*), $(S \cap T)$ (*intersection*), and $(S - T)$ (*set difference*) are valid terms.

3. If S and T are terms, $(S \times T)$ (*Cartesian product*) is also a term. When T is a term and R is a Cartesian product, (R^{-1}) (*inverse*) and (R/T) (*quotient*) are also terms.

(R/T) is the term defined by the relation $((R/T) \times T) = R$, whereas the intuitive meanings of the other compound terms are similar to their meanings in set theory. In fact, the term-oriented language used in NARS is closely related to the formal language used in set theory. The major differences are that in NARS a judgment is no longer binary, and a term is defined both extensionally and intensionally.

With these compound terms, NARS could represent "Tweety is a white bird" as "$Tweety \in (white\text{-}thing \cap bird)$", where the compound term "$(white\text{-}thing \cap bird)$" is the predicate term of the judgment; it could also represent "Mary and John are friends" as "$(\{Mary\} \times \{John\}) \subset friend$", where "$(\{Mary\} \times \{John\})$" is the subject term.

It is easy to extend the grammar defined in Section 3.5 to include these types of compound terms. The study of the related truth-value functions is of course more complex, and is under way.

The extended system not only will handle compound terms provided by the user, but also will be able to generate them by itself. The need for compound terms comes directly from the insufficiency of knowledge and resources. If "white-thing" and "bird" appear together in the intensions and/or extensions of many terms, then treating them as a unit will be much more efficient, hence more intelligent.

What makes this approach different from previous "concept-forming" approaches is: inference in NARS is knowledge-driven. The system does not exhaustively try all possible ways to form compound terms. The term "($\textit{white-thing} \cap \textit{bird}$)" will be formed only when the system finds a term that is both "white-thing" and "bird". After such a compound term is formed, the system will remember its relations with its constituent terms, yet at the same time treat it like other simple terms. Compound terms formed in this way will tend to be quickly forgotten by the system, unless they repeatedly crop up in the system's inference activity. If a compound term survives in this type of competition for resources, the system will come to treat it more and more as a whole, meaning that its relations with its constituent terms will be explicitly considered less and less often. In certain situations, these relations might even be totally forgotten by the system. As a result, in NARS 4 one can expect more complex categorization and inference processes.

- **Long-term limitations**

After the implementation of the new features introduced above, there will of course still be many important and interesting things typical of intelligent systems that

NARS cannot do. The following are some of them.

Higher-order judgments. It is easy to extend the first-order term-oriented language used in NARS 4 to a higher order: we need only allow an inheritance relation, or a judgment about an inheritance relation, to be used as a term. In this way, the system can represent knowledge like “John knows that penguins are birds” and “If Mary is John’s wife, then Ana is Bob’s wife”. What is hard is to set up inference rules for these types of higher-order judgments. It seems that all the old rules (those for first-order judgments) are still valid, but there should be new ones that apply specially to higher-order judgments.

Procedural knowledge. If we allow a term to represent an operation or an event, such as “to tell” or “to move”, then it is possible for the system to represent procedural knowledge, and to use it in planning and scheduling of activities. However, as with higher-order judgments, at the current time it is still not clear what special inference rules are necessary for procedural knowledge. For instance, the system may need rules to organize operations into larger units.

Sensory–motor subsystem. The ability to deal directly with the physical world is not required in my working definition of intelligence, but if a system has some kind of sensory–motor mechanisms, its interface language will *ipso facto* be greatly extended, and it will therefore be more intelligent than a system that has only a symbolic interface. After such an extension, some terms in NARS will no longer correspond to words in a symbolic language, but to perceptual patterns or motor routines of the system that arise in its *physical* (not *verbal*) interaction with its environment.

Natural-language interface. Semantically, the interface language of NARS 3.0 is already more similar to natural language than are those of many other reasoning systems, since the meaning of a term or a judgment is determined by available knowledge, and therefore is fluid and context-sensitive. However, to make NARS handle natural language is still a distant goal. Because we cannot expect NARS to have human experience, it is not clear how far we can go in the attempt to make NARS use our language not only intelligently (which is relatively easy) but also in a human way.

Meta-level self-organizing. All the self-organization in NARS 4 will still be on the level of domain knowledge. In the course of building and modifying its knowledge hierarchy, the system will generate new judgments and new terms, and it will the adjust truth values of judgments and the priorities of chunks, tasks, and pieces of knowledge, but it won't change its own personality parameters (such as the size of a chunk), inference rules (such as how to revise a judgment), or control strategies (such as tampering with its forgetting rate). These kinds of meta-level self-organization require higher-order judgments and procedural knowledge, as well as knowledge about the system itself.

Local axiomatization. Even though I claim that intelligence arises when knowledge and resources are insufficient, this does not contradict the fact that human beings can axiomatize a specific domain ("local axiomatization") by *assuming* sufficient knowledge and resources. If NARS had a similar capacity to locally axiomatize, it would then be able to use numbers and other mathematical notions, make counterfactual assumptions (which involves accepting a statement in the face of negative evidence), design algorithms, and so on, thus achieving greater generality of knowledge and higher efficiency of resources. For pieces of knowledge of this type, we might wish to let their confidence values equal 1, meaning that they are *conventions* made by the system, so will not be directly used to predict future events, and therefore cannot be revised by new evidence.

How to coordinate such *analytical* knowledge with the system's *empirical* knowledge is still an open problem.

Since I do not have clear ideas about how to extend NARS to do these things yet, they will remain limitations of NARS for a long while, but at the same time they will become my goals in future years. However, I still have reason to believe that it makes more sense to tackle these challenges from a non-axiomatic point of view, rather than from a pure-axiomatic or semi-axiomatic point of view, since they are all closely related to the working definition of intelligence — namely, the ability to adapt under insufficient knowledge and resources. Therefore, I feel that what I am now doing constitutes a necessary step toward these goals.

- **Permanent limitations**

There are certain limitations that cannot be removed from the NARS project, since they are fundamentally inconsistent with the working definition of intelligence that gives rise to NARS.

1. Since NARS works with insufficient knowledge and resources, it is impossible for it to have properties that only a pure-axiomatic system can have, such as *consistency*, *completeness*, *decidability*, and so on. When NARS is used to solve practical problems, it cannot guarantee that its results will be *correct* or *optimal*; judgments in NARS are always subject to being revised by the system or refuted by future experience.

2. NARS is not designed to be an accurate model of human reasoning, but to be a reasoning system that has intelligence (according to my working definition of the concept). The system should follow the same *principles* as does the human mind. However, it is not necessary to have the same internal *structure* and *mechanisms* as in the human brain, since computer hardware is fundamentally

different from bio-hardware. Moreover, since NARS' experience will always be different from that of a human being, it is not necessary (though it is still possible to a certain extent) to have the same external behavior as the human mind, such as exactly reproducing some psychological data or passing a certain type of Turing test.

3. NARS is not designed to solve certain domain problems. It is not an expert system nor any other type of computer application system. It is intelligent, not because it can solve problems that no (or few) people can solve (though that might occur in the future), but because it works *in a highly adaptive way.* Like humans, it will make not only intelligent ideas, but also "intelligent" mistakes.

These limitations are easier to deal with than the previous ones — we can just ignore them. This is not to say that the attempt to overcome them is not a valuable goal for research, but simply that such a goal is fundamentally different from (though still related to) our current goal — exploring the essence of intelligence. These limitations of the NARS project mean that if someone is looking for a computer model with these properties, then NARS should not be a candidate, having been designed with other goals in mind.

Bibliography

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

Bai, S. (1991). A mathematical theory for evidence combination. Unpublished manuscript.

Barwise, J. and Perry, J. (1983). *Situations and Attitudes.* The MIT Press, Cambridge, Massachusetts.

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

Bocheński, I. (1970). *A History of Formal Logic.* Chelsea Publishing Company, New York. Translated and edited by I. Thomas.

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

Bonissone, P. (1987). Summarizing and propagating uncertain information with triangular norms. *International Journal of Approximate Reasoning*, 1:71–101.

Bonissone, P. and Decker, K. (1986). Selecting uncertain calculi and granularity. In Kanal, L. and Lemmer, J., editors, *Uncertainty in Artificial Intelligence*, pages 217–247. North-Holland, Amsterdam.

Brachman, R. (1983). What is-a is and isn't: an analysis of taxonomic links in semantic networks. *IEEE Computer*, 16:30–36.

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

Carnap, R. (1952). *The Continuum of Inductive Methods.* The University of Chicago Press, Chicago.

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.

Cherniak, C. (1986). *Minimal Rationality.* The MIT Press, Cambridge, Massachusetts.

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

Dubois, D. and Prade, H. (1982). A class of fuzzy measures based on triangular norms. *International Journal of General Systems*, 8:43–61.

Einhorn, H. and Hogarth, R. (1978). Confidence in judgment: persistence of illusion of validity. *Psychological Review*, 35:395–416.

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

Englebretsen, G. (1981). *Three Logicians.* Van Gorcum, Assen, The Netherlands.

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

Fung, R. and Chong, C. (1986). Metaprobability and Dempster-Shafer in evidential reasoning. In Kanal, L. and Lemmer, J., editors, *Uncertainty in Artificial Intelligence*, pages 295–302. North-Holland, Amsterdam.

Gaifman, H. (1986). A theory of higher order probabilities. In Halpern, J., editor, *Theoretical Aspects of Reasoning about Knowledge*, pages 275–292. Morgan Kaufmann, Los Altos, California.

Good, I. (1965). *The Estimation of Probabilities.* The MIT Press, Cambridge, Massachusetts.

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

Halpern, J. (1990). An analysis of first-order logics of probability. *Artificial Intelligence*, 46:311–350.

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

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

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

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

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, D. (1993). How could a copycat ever be creative? In *Working Notes, 1993 AAAI Spring Symposium Series, Symposium: AI and Creativity*, pages 1–10.

Hofstadter, D. (1995). On seeing A's and seeing As. *Stanford Humanities Review*, 4:109–121.

Hofstadter, D. and the Fluid Analogies Research Group (1995). *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*. Basic Books, New Nork.

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, J., Holyoak, K., Nisbett, R., and Thagard, P. (1986). *Induction*. The MIT Press.

Hume, D. (1748). *An Enquiry Concerning Human Understanding*. London.

Inhelder, B. and Piaget, J. (1969). *The Early Growth of Logic in the Child*. W. W. Norton & Company, Inc., New York. Translated by E. Lunzer and D. Papert.

Keynes, J. (1921). *A Treatise on Probability*. Macmillan, London.

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

Kowalski, R. (1979). *Logic for Problem Solving*. North Holland, New York.

Krantz, D. (1991). From indices to mappings: The representational approach to measurement. In Brown, D. and Smith, J., editors, *Frontiers of Mathematical Psychology: Essays in Honor of Clyde Coombs*, Recent Research in Psychology, chapter 1. Springer-Verlag, Berlin, Germany.

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

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

Kyburg, H. (1974). *The Logical Foundations of Statistical Inference.* D. Reidel Publishing Company, Boston.

Kyburg, H. (1983). The reference class. *Philosophy of Science*, 50:374–397.

Kyburg, H. (1988). Higher order probabilities and intervals. *International Journal of Approximate Reasoning*, 2:195–209.

Kyburg, H. (1992). Semantics for probabilistic inference. In *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, pages 142–148.

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

Lakoff, G. (1994). What is a conceptual system? In Overton, W. and Palermo, D., editors, *The Nature and Ontogenesis of Meaning.* Lawrence Erlbaum Associates, Hillsdele, New Jersey.

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

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

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

Michalski, R. (1983). A theory and methodology of inductive learning. *Artificial Intelligence*, 20:111–116.

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

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

Newell, A. (1990). *Unified Theories of Cognition.* Harvard University Press, Cambridge, Massachusetts.

Nilsson, N. (1991). Logic and artificial intelligence. *Artificial Intelligence*, 47:31–56.

Nosofsky, R. (1991). Typicality in logically defined categories: exemplar-similarity versus rule instantiation. *Memory and Cognition*, 17:444–458.

Paaß, G. (1991). Second order probabilities for uncertain and conflicting evidence. In Bonissone, P., Henrion, M., Kanal, L., and Lemmer, J., editors, *Uncertainty in Artificial Intelligence 6*, pages 447–456. North-Holland, Amsterdam.

Palmer, F. (1981). *Semantics.* Cambridge University Press, New York, 2nd edition.

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

Peirce, C. (1931). *Collected Papers of Charles Sanders Peirce*, volume 2. Harvard University Press, Cambridge, Massachusetts.

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

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

Piaget, J. (1963). *The Origins of Intelligence in Children.* International Universities Press, New York. Translated by M. Cook.

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

Quine, W. V. and Ullian, J. (1970). *The Web of Belief*. Random House, New York.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, 1:81–106.

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

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

Rosch, E. (1973). On the internal structure of perceptual and semantic categories. In Moore, T., editor, *Cognitive Development and the Acquisition of Language*, pages 111–144. Academic Press, New York.

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.

Russell, B. (1901). Recent work on the principles of mathematics. *International Monthly*, 4:84.

Russell, S. and Wefald, E. (1991). *Do the Right Thing*. The MIT Press, Cambridge, Massachusetts.

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

Schweizer, B. and Sklar, A. (1983). *Probabilistic Metric Spaces*. North-Holland, Amsterdam.

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

Shafer, G. (1990). Perspectives on the theory and practice of belief functions. *International Journal of Approximate Reasoning*, 4:323–362.

Simon, H. (1983). *Reason in Human Affairs.* Stanford University Press, Stanford, California.

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

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

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

Touretzky, D. (1986). *The Mathematics of Inheritance Systems.* Pitman Publishing, London.

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

Tversky, A. and Kahneman, D. (1974). Judgment under uncertainty: heuristics and biases. *Science*, 185:1124–1131.

Wang, P. (1986). A reasoning system that can deal with uncertainty. Master's thesis, Peking University, Beijing. In Chinese.

Wang, P. (1992). First ladies and fluid logics. Technical Report 62, Center for Research on Concepts and Cognition, Indiana University, Bloomington, Indiana.

Wang, P. (1993a). 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, P. (1993b). Comparing categorization models: A psychological experiment. Technical Report 79, Center for Research on Concepts and Cognition, Indiana University, Bloomington, Indiana.[1]

Wang, P. (1993c). Non-axiomatic logic (version 2.1). Technical Report 71, Center for Research on Concepts and Cognition, Indiana University, Bloomington, Indiana.

Wang, P. (1993d). Non-axiomatic reasoning system (version 2.2). Technical Report 75, Center for Research on Concepts and Cognition, Indiana University, Bloomington, Indiana.[1]

Wang, P. (1994a). Confidence as higher order uncertainty. Technical Report 93, Center for Research on Concepts and Cognition, Indiana University, Bloomington, Indiana.[1]

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

Wang, P. (1994d). On the working definition of intelligence. Technical Report 94, Center for Research on Concepts and Cognition, Indiana University, Bloomington, Indiana.[1]

Wang, P. (1995a). Grounded on experience: Semantics for intelligence. Technical Report 96, Center for Research on Concepts and Cognition, Indiana University, Bloomington, Indiana.[1]

[1] Available via WWW at *http://www.cogsci.indiana.edu/farg/peiwang/papers.html.*

Wang, P. (1995b). Reference classes and multiple inheritances. *International Journal of Uncertainty, Fuzziness and and Knowledge-based Systems*, 3(1):79–91.

Wang, P. (1995c). An unified treatment of uncertainties. In *Proceedings of the Fourth International Conference for Young Computer Scientists*, pages 462–467, Beijing.

Wang, P. (1996a). Heuristics and normative models of judgment under uncertainty. *International Journal of Approximate Reasoning.* In press.

Wang, P. (1996b). The interpretation of fuzziness. *IEEE Transactions on Systems, Man, and Cybernetics*, 26(4).

Wang, P. and Hsu, C. (1987). A discovery-oriented logic model. In *Second International Conference on Computers and Applications, Beijing, China*, pages 598–604, Beijing. IEEE Computer Society Press, Washington, DC.

Weichselberger, K. and Pöhlmann, S. (1990). *A Methodology for Uncertainty in Knowledge-Based Systems.* Springer-Verlag, Berlin.

Yager, R. (1991). Credibility discounting in the theory of approximate reasoning. In Bonissone, P., Henrion, M., Kanal, L., and Lemmer, J., editors, *Uncertainty in Artificial Intelligence 6*, pages 299–310. North-Holland, Amsterdam.

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

Zadeh, L. (1986). Test-score semantics as a basis for a computational approach to the representation of meaning. *Literary and Linguistic Computing*, 1:24–35.