Non-Axiomatic Reasoning System (Version 2.2)

Pei Wang
Center for Research on Concepts and Cognition
Indiana University
510 N. Fess, Bloomington, IN 47408
pwang@cogsci.indiana.edu

April 14, 1993

Abstract

Non-Axiomatic Reasoning System (NARS) is an intelligent reasoning system, where intelligence means working and adapting with insufficient knowledge and resources.

NARS uses a new form of term logic, or an extended syllogism, in which several types of uncertainties can be represented and processed, and in which deduction, induction, abduction, and revision are carried out in a unified format. The system works in an asynchronously parallel way. The memory of the system is dynamically organized, and can also be interpreted as a network.

After present the major components of the system, its implementation is briefly described. An example is used to show how the system works. The limitations of the system are also discussed.

1 Introduction

Non-Axiomatic Reasoning System (NARS) is an intelligent reasoning system.

Intelligence is understood here as *the ability of working and adapting to the environment with insufficient knowledge and resources.* More concretely, to be an information processing system that works under the *Assumption of Insufficient Knowledge and Resources (AIKR)* means the system must be, at the same time,

a finite system — the system's computing power, as well as its working and storage space, is limited;

a real-time system — the tasks that the system has to process, including the assimilation of new knowledge and the making of decisions, can emerge at any time, and all have deadlines attached with them;

an ampliative system — the system not only can retrieve available knowledge and derive sound conclusions from it, but also can make refutable hypotheses and guesses based on it when no certain conclusion can be drawn; and

an open system — no restriction is imposed on the relationship between old knowledge and new knowledge, as long as they are representable in the system's interface language.

Furthermore, to be an *adaptive system* (or *learning system*) means the system must also be

a self-organized system — the system can accommodate itself to new knowledge, and adjust its memory structure and mechanism to improve its time and space efficiency, under the assumption that future situations will be similar to past situations.

By *reasoning system*, I mean an information processing system that has the following components:

a formal language for the communication between the system and the environment;

an interpretation of the formal language that makes its sentences correspond (maybe loosely) to human knowledge represented in natural language;

a user interface through which the system can accept knowledge from the environment, and answer questions according to its knowledge;

an inference engine with some inference rules to carry out tasks, such as match questions with knowledge, generate conclusions from promises, and derive subquestions from questions;

a memory that store the tasks to be processed, and the knowledge according to which the tasks are processed; and

a control mechanism that is responsible for the choosing of premise(s) and inference rule(s) in each step of inference, and the maintaining of the memory.

In this paper, I will describe NARS-2.2's theoretical foundation, its various components, its implementation, and its limitations. I'll briefly compare it with other systems, but leave detailed discussions to future papers.

2 Theoretical Foundation

What differentiates NARS from other AI approaches in theoretical consideration is its working definition of intelligence: the ability to work and adapt with insufficient knowledge and resources.

Of course it is too early to make a universally acceptable definition for intelligence, but it is still necessary for each research paradigm to set up its working definition, since such a definition can specify the goal of the research, and provide a kernel for the whole theory and technical tools.

In the following, I will explain why the NARS project (previously referred as Non-Axiomatic Logic, or NAL) chooses such a working definition for intelligence.

2.1 The working environment of human mind

It is easy to see that the human mind often works under AIKR, and (of course) is an adaptive system:

- Its computing power, as well as its working and storage space, is limited. We cannot think about too many things simultaneously. We cannot remember everything we knew previously.
- It has to work in real time. Many information-processing tasks are imposed upon us by our environment, in spite of whether we are ready to process them; all the tasks have deadlines attached (also determined by the environment), regardless whether or not we can get a satisfying result by that deadline.
- It has to guess. If we only acted on what we are absolutely sure about, we could hardly do anything at all. To guess and then make mistakes is still better than to do nothing.
- It is open to new knowledge. We cannot make restrictions on what can happen in the future. Even if there is something happening that conflicts with our current beliefs, we cannot refuse to accept it as new evidence.
- By adapting to its environment, a human mind makes self-adjustments based on the current situation and previous experience. Therefore, only under the assumption that the future situation will be (more or less) similar to past situations (at certain general level), can such adjustments be expected to really improve the system's behavior.

I don't mean that we *never* work with sufficient knowledge and resources. For some familiar and simple problems, we can assume that our knowledge and resources are relatively sufficient for solving them. For such problems, an information-processing model that assumes sufficient knowledge and resources is a good approximation, while considering AIKR will make things unnecessarily complicated. My point here is: it makes sense to study adaptive systems under AIKR, since the human mind often (though not always) works in this way. To us, the available information is usually too much to be processed, at the same time too little to make infallible predictions.

2.2 The assumptions of traditional theory

In mathematics, things are extremely different. There what we have are idealized objects with idealized relations among them. Mathematicians (like Hilbert) hope to build *a consistent, complete and decidable system* (I call it a *full-axiomatic* system). It has a set of axioms and a set of derivation rules, and all questions in the domain can be answered by theorems derived from the axioms, following an algorithm. In such a case, and only in such a case, the sufficiency of knowledge and resources can be completely achieved in the following sense:

- There is no upper bound on the number of axioms, theorems, and rules that the system can maintain at a given time;
- The system answers questions in a one-by-one way, and the turn-around time is determined by the complexity of the algorithm;
- The system has complete knowledge about the domain, so no guessing is necessary, that is, all answers are guaranteed to be true;
- The system already knows everything it needs to know at the very beginning, therefore it is no longer open to new knowledge;
- Such a system has no need to adapt or to learn, since it is already good enough for the given domain.

Computer science, with its close historical relations with mathematics, inherits the assumption of the sufficient knowledge and resources. This assumption is behind the definition of *Turing machine*, *computation*, *algorithm*, as well as other kernel concepts of the discipline. As a result, in a domain where we have sufficient knowledge and resources (in respect to the problem to be solved), we have a solid theoretical foundation and many useful tools. In contrary, if we have to solve problems under AIKR, there is no ready-made theory to guide us. It is another reason to study adaptive systems under AIKR, since such systems haven't been adequately studied yet.

2.3 The problems of artificial intelligence

Only when scientists begin to explore artificial intelligence, did the systems designed according to traditional computer science theory begin to reveal their weaknesses: they are too rigid, too brittle, not creative, and so on ([Smolensky 88] and [Holland 86]). Even after such systems have solved some problems which are hard or unsolvable for a human being, we still don't think they are intelligent. It seems, as stated in *Tesler's Theorem* ([Hofstadter 79]), "AI is whatever hasn't been done yet".

This suggests that in our everyday usage, "intelligence" does not merely refer to *the ability of solving certain problems*, but to *the (meta)ability to solve problems under certain constraints*: the solutions should be creative, context-dependent, flexible, efficient, robust, and so on. If a problem is solved by following a predetermined algorithm, even though the solution itself may be excellent, the solving process is not considered intelligent. On the other hand, the designing of the algorithm will be considered as intelligent, as long as it is not done by following a meta-algorithm.

This observation supports our previous claim: only under AIKR, intelligence becomes *possible* and *necessary*. We can rephrase *Tesler's Theorem* as: "AI is whatever hasn't been done by a full-axiomatic system yet".

Let's take a look at a list of subfields of AI:

understanding natural language
producing natural language
recognizing scenes
recognizing sounds
planing actions
playing games
proving theorems
thinking creatively
thinking analogically
......

and compare the list with other computer application domains, such as:

evaluating formulas
maintaining databases
controlling assembly lines
......

We can see that the problems in the first list, which are relatively easy for human beings but hard for computers, are all inherently related to AIKR. If we abstract and idealize these problems to such an extent as to assume the knowledge and resources are sufficient, then the problems will either become drastically different from the original ones, or even completely disappear. For example, if knowledge and resources are sufficient, many problems in game playing and theorem proving can be solved by exhaustive search, while creative thinking become impossible and analogical thinking unnecessary — only deductive thinking is left.

2.4 AI and AIKR

As a matter of fact, to design a computer system which can work and adapt with insufficient knowledge and resources is not a completely new idea to the discipline of artificial intelligence and cognitive science, since

1. some sub-fields, such as machine learning and uncertain reasoning, are totally devoted to such efforts;
2. many theories/systems assume certain types of insufficiency of knowledge and/or resources; and
3. AIKR is closely related to ideas like *minimal rationality* in [Cherniak 86] and *limited rationality* in [Russell and Wefald 91].

What distinguishes the NARS project from other approaches is:

1. here "adapting under AIKR" is accepted as a *whole*,
2. the principle is further specified, and
3. it is implemented in a reasoning system.

The following are some anticipated objections and my responses:

The "idealization" objection: "We know that the human mind usually works under AIKR, but if you want to set up a formal model, you must somehow idealize the situation."

Response: It is true that any formal model is an idealization, and so is NARS. The problem is what to omit and what to preserve in the idealization. In NARS-2.2, many factors that influence human reasoning are ignored, but AIKR is strictly assumed throughout. Otherwise, the system would work in a different manner: it could still solve many practical problems, but in a not-so-intelligent way, like many "expert systems".

The "goodness" objection: "There will be many defects for a system to work under AIKR. The system will provide bad answers to us."

Response: This is absolutely true. In fact, we'll see that NARS makes various types of mistakes. However, this cannot be used as an objection to AIKR, but should be understood as a price we have to pay. From our working definition of intelligence, it follows that an *intelligent* system need not necessarily be better than an *unintelligent* one in terms of solving practical problems, such as addition ([Hofstadter 79]). Intelligence is what we have to use when we cannot solve a problem in a mechanical (full-axiomatic) way. Therefore, with all its defects, to work under AIKR is still better than to do nothing at all, or to "idealize" the original problem into a totally different one, then solve the latter perfectly.

The "simplification" objection: "AIKR is necessary, but at the beginning, we should decompose it into separate problems, then solve them one by one to make things easy."

Response: This is the approach that many current AI approaches take. They try to modify or extend some of the assumptions of traditional theory, and keep the others untouched. I call them *semi-axiomatic* approaches. For example, Bayes networks and Non-monotonic logic systems are only open to certain types of new knowledge, many learning systems still use binary logic, few systems work in real time, and so on. As a result, the applicability of the theory is more limited. On the other hand, the theoretical assumptions conflict with each other, and a lot of efforts have to be made to deal with the inconsistency. As an example, in many non-monotonic logic system, a concrete conclusion (such as "Tweety can fly") can be rejected by new evidence (such as "Tweety is a penguin"), but a "default rule" (such as "Birds normally can fly") cannot ([Reiter 87]). Is it possible for us to draw such a line in our empirical knowledge? In NARS, since AIKR is treated as a whole, the theoretical foundation of the system is more consistent, and (amazingly) the system is simpler technically. It seems that it is better to treat "adapting under AIKR" as *one* problem with many aspects than as many loosely related problems.

The "importance" objection: "AIKR should be respected, but it is not so important as to be referred to as AI's *kernel*. AI's kernel should be ⋯ ⋯" (there will be hundreds of different opinions).

Response: What we expect from a working definition is: it is relatively simple, but has plentiful implications. I don't claim that "adapting under AIKR" can explain all phenomena that are usually associated with intelligence, but it does explain many of them. In the following, I'll try to derive non-monotony, parallel processing, distributed representation, meaning fluidity, fault-tolerance, internal competition, autonomy, context-sensitivity, and so on, from such a working definition of intelligence. Therefore, it is possible for us to have a pretty small kernel that explains many phenomena. In this way, we can also have a better idea about the relations among these phenomena. By the NARS project, I want to explore how far this working definition can lead us. I hope it will lead us further than other current definitions of intelligence do.

3 Interface Language and Its Interpretation

As a reasoning system, NARS needs a formal language to communicate with the environment. Its environment is either other information processing agents (human or computer), or, when it is a subsystem of a system, other sub-systems (sensory subsystem, motor sub-system, natural language interface, and so on). NARS accepts new knowledge (told by the environment) in the language, and answers questions (asked by the environment) in the language, too.

3.1 Term-oriented language

Traditionally, the meaning of a formal language $\mathbf{L}$ is provided by a model-theoretic semantics, where a term a in $\mathbf{L}$ indicates an object A in a domain $\mathbf{D}$, and a predicate P in $\mathbf{L}$ indicates a property p in $\mathbf{D}$. A proposition $P(a)$ is true if and only if the object A actually has the property p.

Such a semantics works well for mathematics (or generally, for all full-axiomatic systems and some semi-axiomatic systems), but due to AIKR, it is no longer applicable to a non-axiomatic system. With insufficient knowledge, it is impossible to measure to what extent a proposition can match the *real world*. Even the concept of *possible world* (or *state-description*, as in [Carnap 50]) is too ideal to be used here: we cannot list all possible worlds, since to do that we have to list all possible terms and all possible predicates. If it could be done, then the system would be no longer open to new knowledge that including novel terms and/or predicates.

What makes the things more complicated is:

1. the knowledge provided by the environment may be uncertain or conflict with each other;
2. when knowledge is insufficient, the system has to guess; and
3. if there are more than one reasonable guess, the system have to decide which is more strongly supported by the system's past experience, therefore *more plausible* under the assumption that the future will be similar to the past.

It still makes sense to talk about the *truth value* of a proposition under AIKR, since the system need to indicate how the proposition is supported/refused by what it was told by the environment in the past. Such a truth value is only indirectly related to the "real world". After all, for NARS to be intelligent, what it need to do is to adapt to *its environment*, which, if consists of sensory/motor sub-systems or honest human beings, will lead the system to set up a model for the real physical world, but it is not necessarily the case.

As a result, binary logics, even three-valued logics or model logics, are not suitable here — we need more information. However, under AIKR, it is impossible to explicitly record all pros and cons for each *statement* S (that is, a sentence of the interface formal language) and to compare them in general, so we have to measure them in some way, and to compress the information into numbers. For a statement S, if we can find a natural way to distinguish the *positive evidence* K_S^+ and the *negative evidence* K_S^- for it from all available knowledge K, and to define a measurement function F for the *amount* of evidence, then it seems that the simplist way is to use the pair $< F(K_S^+), F(K_S^-) >$ (or $< F(K_S^+), F(K_S) >$, where $F(K_S) = F(K_S^+) + F(K_S^-)$ is the amount of all evidence) as S's *current* truth value in the system.

However, if we try to apply this idea to extend the truth value of first order predicate logic, we'll meet Hempel's famous *Ravens Paradox* ([Good 83]): A green tie, or white shoe, is positive evidence for "All ravens are black", since it is positive evidence for the logically equivalent statment "All non-black things are non-ravens". But this is counter-intuitive, and it will imply other strange conclusions, such as a green tie is also positive evidence for "All ravens are red", "All ravens are blue", and so on. It is unnecessary and impossible to discuss the paradox in detail here, so I only say that this is one reason for NARS to abandon predicate logic and to use *term logic.*

In traditional term logics, such as Aristotle's and Peirce's, all statement consists of two *terms*, a *subject* S and a *predicate* P, which are related by the copula *be*, which is interpreted as a whole-part relation, as *including*, or *belonging to.*

In NARS, the "be" relation is represented as "$\subset$", and interpreted as *inheritance.* A statement "$S \subset P$" can be intuitively understood as "S is a kind of P" (from an *extensional* point of view) and "S has P's property" (from an *intensional* point of view). Or correspondingly, we can say "S inherits P's attributes" and "P inherits S's instances", which can be further described as "if $X \subset S$, then $X \subset P$" and "if $P \subset Y$, then $S \subset Y$" , respectively (that is why the relation is called an inheritance relation). This interpretation is more general than the pure extensional "including" interpretation.

Under such an interpretation, we can see that the positive evidence for "$S \subset P$" consists the common instances and common properties of S and P, while the negative evidence consists of S's instances which are not shared by P and P's properties which are not shared by S. In such a system, a green tie is no longer a piece of evidence for "Ravens are black-thing", and the negating of a term (to get "non-raven" or "non-black-thing") is not even a valid operation.

3.2 Truth value

Now the measurement function F mentioned above can be intuitively defined as *counting*: Ideally, if the system checked S's instances n times in past, and in m times the instance turned out also to be in P, then $F(K_S^+) = m$, and $F(K_S) = n$. Symmetrically, if it checked P's properties n times, and in m times the property happened to be S's, too, then $F(K_S^+) = m$, and $F(K_S) = n$.

The "counting" form of truth value is good for certain purpose, but it is not always make sense to test a statement in such a way, and we often prefer a relative measurement as (extended) truth value, such as some kind of probability. It is easy to see that from above discussion we can use m/n as a relative measurement, which is the *frequency* of successful inheritance between the subject and predicate. We'll refer it as f, and take its value in [0,1].

Though f can be used to compare which statement comparatively gets more positive support from available knowledge, it provides no information for how the ratio should be modified when future evidence comes. For this reason, we need another value to measure the system's *confidence* about the current frequency of the statement, that is, how stable it is in the future. Under AIKR, it make no sense to estimate the range that f can be in the infinite future — it can be anywhere in [0,1], since there is no limitation on the amount of coming positive or negative evidence. But it is still make sense to talk about the *near future*, that is, when a constant amount of new evidence is coming. Therefore, the *confidence* of a statement (referred as c) is defined as *the ratio of the current amount of evidence to the future amount of evidence*, that is: $c = n/(n+k)$. Here k is a constant in the system, indicating that by "new evidence in near future" it means to check the inheritance relation for k more times ($k > 0$, but not necessary to be an integer). We say this ratio can be used as a measurement of confidence, since if c is pretty big, it means we have tested the statement many times, therefore our opinion about it is quite stable, no matter what will happen in the near future.

In this way, it is possible to provide a unified representation and interpretation for the various types of uncertainties ([Bhatnagar and Kanal 86]): *randomness* mainly corresponds to estimate f from an extensional point of view (by considering common instances of the two term); *fuzziness* (or *representativeness*, *typicality*) mainly corresponds to estimate f from an intensional point of view (by considering common properties of the two term); and *ignorance* corresponds to the estimation of c.

Moreover, we can exactly indicate the interval that f can be in the near future: if all of the k tests give negative evidence, f will be as low as $a = m/(n+k)$; if all of the k tests give positive evidence, f will be as high as $z = (m+k)/(n+k)$. Anyway, it will be in the interval $[a, z]$ in the near future. Defined in this way, we get $z - a = k/(n+k) = 1 - c$, which tell us that the confidence c can also be understood as the complement (to 1) of the width of the interval: the narrower the interval, the more confident the system is about the frequency of the statement.

Now, we have three different forms of truth value, and there are one-to-one mappings among them, which are shown in Table 1.

To have different, but closely related forms and interpretations for truth value have many advantages:

- It give us a better understanding about what the truth value really means in NARS, since we can

	$\{m, n\}$	$< f, c >$	$[a, z]$
$\{m, n\}$		$m = k\frac{fc}{1-c}$ $n = k\frac{c}{1-c}$	$m = k\frac{a}{z-a}$ $n = k\frac{1-(z-a)}{z-a}$
$< f, c >$	$f = \frac{m}{n}$ $c = \frac{n}{n+k}$		$f = \frac{a}{1-(z-a)}$ $c = 1 - (z - a)$
$[a, z]$	$a = \frac{m}{n+k}$ $z = \frac{m+k}{n+k}$	$a = fc$ $z = 1 - c(1 - f)$	

Table 1: Different forms of truth value and their relationships

explain it in different ways. The mappings also tell us the interesting relations among the various ways of uncertainty measurement.

- It provides a user-friendly interface: if the environment of the system is human users (as the case of NARS-2.2), the uncertainty of a statement can be represented in different forms, such as "I've tested it n times, and in m of them it was true", "Its past success frequency is f, and its stability in the near future is c", or "I'm sure that its success frequency with remain in the interval $[a, z]$ in the near future". Using the mappings in the above table, we can maintain an unique truth value form as internal representation (in NARS-2.2, we use the $< f, c >$ form), and translate the other two into it in the interface.

- It make the designing of inference rules easier. For each rule, there are functions calculating the truth value of the conclusion(s) from the truth values of the premises, and different functions correspond to different rules. For some rule, it is easier to choose a function if we treat the truth values as "countings", while for another rule, we may prefer to treat them as "ratios" or "intervals". No matter what form and interpretation is used, the *information* carried is actually the same.

- It is easier to compare the NARS approach to various other approaches for uncertain reasoning (see [Bhatnagar and Kanal 86] and [Spies 89]), such as f vs. *probability* ([Pearl 88]), f vs. *degree of membership* ([Zadeh 85]), c vs. *higher-order probability* ([Paaß91]), $[a, z]$ interval vs. *probability interval* ([Weichselberger and Pöhlmann 90]), $[a, z]$ interval vs. $[Bel, Pl]$ interval ([Shafer 76]), and so on. The comparisons cannot be done in this paper, and I only want to point out that all the approaches mentioned above make various constraints on their working environment, which assume certain kind of sufficient knowledge or resources. According to previous discussion, there approaches lead to semi-axiomatic systems.

3.3 Degree of belief

All of the above defined measurements in NARS are completely based upon the system's past experience, without any assumption about what will happen in the future. However, we often need to choose among competing judgments in terms of which is most likely to be true in the future, or to bet on their truth. For models that use a single number to represent the uncertainty of a judgment, such as probabilistic logic and fuzzy logic, it is simple — we can simply compare the relevant numbers. However, for NARS, we need to somehow summarize the pair of numbers (whether as $\{m, n\}$, $< f, c >$, or $[a, z]$) into a new measurement, and use it to represent prediction. Let's call it *degree of belief*, and referred as d.

Though success frequency is important for prediction, we cannot simply use f as d, since f may be unstable. If we have only tried an experiment once, and it succeeded, usually we are not 100 percent sure that it will succeed again when repeated. Intuitively, d can be defined as the expectation of f in the near future, because in this way we also take its possible variance into account. Since no matter what happens, f will be within $[a, z]$, therefore it is natural for us to use the unbiased expectation $(a + z)/2$ as d's value.

Using the mappings among different forms of truth value, we can get some interesting results about d.

When d is represented in the "ratio" form, we get

$$c = \frac{d - 0.5}{f - 0.5}$$

Since $c < 1$, d seems always more "conservative" (more close to the middle point) than f, while c indicates the ratio that f is "compressed" to the middle point to become d. When c is very small, d will be within the neighborhood of 0.5, in spite of what f is — corresponding to the "little evidence" situation; when c is very big, d will be within the neighborhood of f — corresponding to the "much evidence" situation, where f, d, a, and z are almost equal to each other. However, different from probability theory, in NARS it is not assumed that there are limitations for f or d when the system get more and more knowledge. On the other hand, it is not assumed that the system's beliefs are consistent.

When d is represented in the "counting" form, we get

$$d = \frac{m + \frac{k}{2}}{n + k}$$

This formula, when $k = 2$, leads us to the famous "Law of Succession" of Laplace, that is, after m successes in n trials, the probability of the next success should be $(m+1)/(n+2)$ ([Good 83]).

If we let our k parameter (that is, what we mean by "the near future") vary, what we get is a continuum, which is very similar to Carnap's λ-continuum ([Carnap 52]). The bigger the k is, the more "cautious" the system is — it takes more evidence to reach a certain level of confidence. Therefore, k is one of the *personality parameters* of the system.

These relations suggest that this approach of uncertainty measurement and interpretation may lead us to a unification of some other approaches that come from different motivations, and this work is closely related to, but not included by, the previous study of probability theory and inductive logic.

3.4 NARS-2.2's grammar

In NARS-2.2, there is another interpretation about the inheritance relation "be" between two terms, which intuitively corresponding to the membership relation "$\in$" in set theory, with an extension of its interpretation to including intensional factors. In NARS, the statement "$S \in P$" can be identically rewritten as "$\{S\} \subset P$". In this way, we can distinguish "to be a kind of" and "to be a member of". As a result, NARS can handle singular terms, which were omitted by Aristotle ([Lukasiewicz 51]). To make things simple, we can transfer between the "$\in$" relation and the "$\subset$" relation in the user interface of the system, and only have the "$\subset$" relation in the internal representation. Now, in NARS-2.2, a term can be a word or a singleton set of a term.

A *judgment*, as the basic unit of knowledge, is a statement with an attached truth value. Each judgment represents the system's opinion about to what extent a term *can be used as* another term, and all the reasonings in the system are about the "can-be-used-as" relation. In this way, NARS shares some common spirit with the Copycat project ([Hofstadter and Mitchell 92]) and the Tabletop project ([Hofstadter and French 92]).

NARS-2.2 accepts two types of *questions* from its environment:

1. "Yes/no" question (or "recognition"), which has the form of a statement. To answer such a question means to give it a truth value with the highest *confidence*, according to the knowledge of the system. In other words, the system is asked to assign a (as stable as possible) truth value to the given statement.

2. "What" question (or "recall"), which has the form of a statement where one of the two terms is replaced by a question mark. To answer such a question means to find a term to substitute the question mark, which will make the statement has the highest *degree of belief*, according to the knowledge of the system. In other words, the system is asked to make a best guess about which term can make the statement true.

Finally, Table 2 shows the formal grammar of NARS-2.2's interface language, where the *truth-value* can be represented in three ways:

1. $\{m, n\}$, where m is a non-negative real number, n is a positive real number, and $n \geq m$;

2. $< f, c >$, where $f \in [0, 1]$, and $c \in (0, 1)$; or

3. $[a, z]$, where $0 \leq a < z \leq 1$, and $1 > z - a$.

Beyond these valid truth values, there are two limitation points that need to be mentioned:

Null evidence: This is represented by $n = 0$, or $c = 0$, or $z - a = 1$. It means that the system actually know nothing at all about the statement;

$$
\begin{array}{rcl}
<\textit{judgment}> & ::= & <\textit{statement}><\textit{truth-value}> \\
<\textit{question}> & ::= & <\textit{statement}> \mid <\textit{query}> \\
<\textit{query}> & ::= & ?<\textit{be}><\textit{predicate}> \mid <\textit{subject}><\textit{be}>? \\
<\textit{statement}> & ::= & <\textit{subject}><\textit{be}><\textit{predicate}> \\
<\textit{subject}> & ::= & <\textit{term}> \\
<\textit{predicate}> & ::= & <\textit{term}> \\
<\textit{term}> & ::= & <\textit{word}> \mid \{<\textit{term}>\} \\
<\textit{word}> & ::= & <\textit{letter}> \mid <\textit{letter}><\textit{word}> \mid <\textit{word}> \text{-} <\textit{word}> \\
<\textit{letter}> & ::= & a|b|\cdots|y|z \\
<\textit{be}> & ::= & \subset \mid \in
\end{array}
$$

Table 2: Grammar of NARS-2.2's interface language

Total evidence: This is represented by $n \to \infty$, or $c = 1$, or $z = a$. It means that the system already know everything about the statement — no future modification of the truth value is possible.

Under AIKR, there is no need for the system to represent the first type of "knowledge", and impossible to have the second type. But, as limitation points, they can help us to design and understand truth values and related functions, and they are useful in future extensions of NARS.

4 Inference Rules

By inference, the system can modify its knowledge, derive new knowledge and answer questions according to available knowledge.

In NARS-2.2, there are two major types of inferences: forward (from two judgments to a new judgment) and backward (from a question and a judgment to an answer or a new question). In both cases, the two premises (two judgments, or one judgment and one question) must share a common term to get a conclusion (judgment or question). Due to AIKR, the conclusions in each step of inference are only based on the information provided by the premises, therefore may be revised or refused in the future.

4.1 Revision

Under AIKR, it is possible (actually it is usually the case) for NARS to be inconsistent in the sense that at a certain time, there are two co-existent judgments which share the same statement, but attach different truth values to it. One reason for this to happen is: due to insufficient knowledge, old knowledge may conflict with new knowledge. Another reason is: due to insufficient resources, the system cannot find all potential conflicts among its current knowledge.

It is not to say that such inconsistency is not processed. In NARS, inconsistent judgments have the following effects:

- unlike in first order predicate logic, where any conclusion can be derived from a pair of propositions which only differ in there truth values, in NARS an inconsistency is a local problem that not all results are affected;
- as soon as such an inconsistence is found by the system, a revision rule is used to solve it locally, that is, to get a conclusion that summarize the two premises.

If the two premises are

$$S \subset P\{m_1, n_1\}$$

$$S \subset P\{m_2, n_2\}$$

(where the truth values are represented in the counting form), and the two premises are *independent* to each other (that is, no evidence is repeatly counted in the two premises), then it is natural to let the conclusion be

$$S \subset P\{m_1 + m_2,\ n_1 + n_2\}$$

. In this way, the system can resolve inconsistence, and accumulate evidences about a statement gradually and incrementally.

Independence between judgments in NARS-2.2 is defined by the following three principles:

1. All *input judgments* (provided by the environment) are independent to each other, as long as they enter the system at different times.

2. If judgment J_1 is derived from the set of input judgments S_1, and J_2 is an input judgment, then J_1 and J_2 are independent to each other if and only if J_2 isn't in S_1.

3. If J_1 and J_2 are derived from S_1 and S_2 (both are set of input judgments) respectively, then J_1 and J_2 are independent to each other if and only if S_1 and S_2 have no common element.

According to the definition, it is necessary to record a complete derivation history for every judgment to determine whether two arbitrary judgments are independent to each other, but this will violate AIKR. As a result, only a constant part of the "family tree" can be recorded for each judgment, and it is impossible to accurately indicate each input judgment's contribution to the truth value of the judgment at hand. That means the system's decisions about independence of the premises are not always correct, and when two judgments are not independent to each other, it is hard to say how the truth values should be modified. It is also true for human beings: we cannot always correctly recognize and handle correlated evidence.

There are two more issues should be mentioned:

(1) As long as two judgments are independent, they are merged, no matter *how* the truth values are estimated in them: either by directly comparing the properties or instances of the two terms, or by some indirect methods. As a result, the truth value, when represent in the counting form $\{m, n\}$, is only used as a measurement — it doesn't say the system has actually tested the statement n times, and it was true in m of them. What it means is: the system's belief about the statement is *as strong as* it has been tested for n times, and it was true in m of them. On the other hand, evidence from property comparisons and instance comparisons are usually mixed, since they are equally useful to suggest a future inheritance relation between the two terms. This will provide an explanation for the results of some psychological experiments, such as why people often do not clearly distinguish representativeness from probability ([Tversky and Kahneman 74]).

(2) Since what the system counted is *how many times* a term can be used as another according to the system's past experience, repeated input will be considered as independent, even if they have the identical form, and have (known or unknown) common source in the environment. In fact, the truth value of a statement is about the frequency of certain "psychological events" (the extent that one term can be used as another), rather than about certain "physical reality". For example, to some Australian, the judgment "Swans are black" may have a higher frequency than "Swans are white", even though they know that globally and physically, most swans are white. If they are not writing a report in ornithology, they have good reason to base their knowledge on their personal environment.

4.2 Syllogism

As a term logic, the basic inference rule of NARS is a extended syllogism.

When two premises share exactly one term, there are three cases:

1. the shared term is the subject of one premise and the predicate of the other,

2. the shared term is the subject of both premises,

3. the shared term is the predicate of both premises.

Following Peirce's definition, they are referred as *deduction*, *induction*, and *abduction*, respectively.

4.2.1 Deduction

Deduction is the way that the inheritance relations are transferred by the means of the shared term. Formally, there are two premises

$$M \subset P < f_1, c_1 >$$

$$S \subset M < f_2, c_2 >$$

and from them

$$S \subset P < f, c >$$

is inferred as a conclusion.

To determine the value of f and c, we can set up several constraints based on our interpretation about the truth values:

1. When and only when $f_1 = f_2 = 1$, $f = 1$. That means the conclusion have no negative evidence if and only if the two premises have no negative evidence.

2. When and only when f_1 or f_2 is 0, $f = 0$. That means if all evidence for a premise is negative, no positive knowledge can be gotten for the conclusion.

3. When $f_1 = f_2 = 0$, $c = 0$. That is, from two totally negative premises "M is not P" and "S is not M", we get no information about whether S and P have shared instances or properties.

4. $c \leq \min\{c_1, c_2\}$. That is, the confidence of the conclusion is less than either of the premise's.

5. When $f_1 = c_1 = 1$, $< f, c > = < f_2, c_2 >$, and symmetrically, when $f_2 = c_2 = 1$, $< f, c > = < f_1, c_1 >$. That means, when $f_1 = c_1 = 1$, there is a "ideal inheritance" relation between M and P. As a result, P completely inherits M's relation with S.

6. When and only when $c_1 = c_2 = 1$ and one of f_1 and f_2 is 1, $c = 1$. That means, for the conclusion to approaching "total evidence", there must be an "ideal inheritance" in the premises, and the other premise must also approaching "total evidence".

In NARS-2.2, the following functions are used, which satisfy the above constraints:

$$\begin{aligned} f &= f_1 f_2 \\ c &= (f_1 + f_2 - f_1 f_2)\, c_1 c_2 \end{aligned}$$

Deductions in term logics (such as in NARS) differs fundamentally from those in predicate logics (such as in first order predicate logic). In predicate logic, whether a premise can imply a conclusion is totally determined by their truth values. This is the root of the infamous *implication paradox*: a true proposition can be derived from any proposition, and any proposition can be derived from a false proposition. On the contrary, deductions (as well as inductions and abductions) in a term logic require the two premises to share a common term, therefore the premises and the conclusion must be semantically related. On the other hand, the semantic relation is detected and processed syntactically by checking for a common term.

4.2.2 Induction

Induction is the way that the inheritance relations are established by comparing the instances (represented by the shared term) of the two unsheared terms. Formally, there are two premises

$$M \subset P < f_1, c_1 >$$

$$M \subset S < f_2, c_2 >$$

and from them

$$S \subset P < f, c >$$

is inferred as a conclusion.

According to the interpretation of the truth value, we know that M can contribute to the conclusion no more than be counted as an "unit evidence" (that is, to make $n = 1$ for the conclusion). To what extent M is a piece of evidence for the conclusion depends on whether the premises have a high confidence value, and whether M is really in S. If M is not in S, then it is not a related evidence for "$S \subset P$". (A green tie is not a piece of evidence for "Ravens are blackthing".) Therefore, I choose $n = f_2 c_1 c_2$. From the mappings in Table 1, we can determine c form n by the formula $c = n/(n + k)$.

For f, it seems the most natural guess is f_1: here we generalize M's relation with P to S's relation with P.

As a result, we get

$$\begin{aligned} f &= f_1 \\ c &= \tfrac{f_2 c_1 c_2}{f_2 c_1 c_2 + k} \end{aligned}$$

Obviously, by interchanging the premises, we can get a symmetric inductive conclusion

$$P \subset S < f', c' >$$

. Generally, inductive conclusions are generated in pairs, but usually with different truth values.

Inductions in NARS has a distinctive feature: the induction rule not only *generate* the inductive conclusion, but also *evaluate* it at the same time, while in previous study of induction, both in the field of machine learning and in the field of inductive logic, the generating (or *searching, discovering*) and the evaluating (or *conforming, testing*) of inductive hypotheses are traditionally treated as different phrases of a induction procedure, and carried by different mechanisms ([Michalski 83] and [Carnap 50]).

4.2.3 Abduction

Abduction is the way that the inheritance relations are established by comparing the properties (represented by the shared term) of the two unsheared term.

Formally, there are two premises

$$P \subset M < f_1, c_1 >$$

$$S \subset M < f_2, c_2 >$$

and from them

$$S \subset P < f, c >$$

is inferred as a conclusion.

In NARS-2.2, inheritance relations about properties and inheritance relations about instances are treated in a completely symmetric way. As a result, we can directly get the functions for abduction from the functions for induction:

$$\begin{array}{rcl} f & = & f_2 \\ c & = & \frac{f_1 c_1 c_2}{f_1 c_1 c_2 + k} \end{array}$$

"Abduction" in NARS has a different meaning from what the word means in the field of machine learning. Originally, abduction was defined by Peirce as one of the three types of syllogism (as in NARS), but it was described as "only justified by its *explaining* an observed fact" ([Peirce 32]). Later, as predicate logic replaced term logic as the dominant logical language, "abduction" has been used to refer to explaining reasoning in predicate logic, as the case in machine learning study ([Falkenhainer 90] and [Michalski and Kodratoff 90]).

Since k is a positive constant, the confidence of a conclusion gotten directly from abduction (or induction) is less than $1/(1+k)$ (in NARS-2.2, k is 2, so $c < 1/3$), while the confidence of a deductive conclusion can have 1 as limitation. Therefore, the generally accepted difference between deduction and induction/abduction still exists in NARS: as *ampliative reasoning*, inductive/abductive conclusions are usually weaker (less confident) than deductive conclusions ([Peirce 32]), that means they are more likely to be changed by new evidence. These conclusions can become stronger by merging with consistent judgments through revision. For example, from "Doves are birds" and "Doves are flyers", the system can induce "Birds are flyers", but with a low confidence. Then, the system can get a similar conclusion from "Swans are birds" and "Swans are flyers". When the two "Birds are flyers" meet, they will merge into a stronger (with a higher confidence) "Birds are flyers", since the two premises are independent to each other (See **Section 8** and **Appendix** for details).

$J_2 \setminus J_1$	$M \subset P(t_1)$	$P \subset M(t_1)$
$S \subset M(t_2)$	$S \subset P(F_{ded})$	$S \subset P(F_{abd})$ $P \subset S(f_{abd})$
$M \subset S(t_2)$	$S \subset P(F_{ind})$ $P \subset S(f_{ind})$	$P \subset S(f_{ded})$

Table 3: Extended syllogism

4.2.4 The new syllogism

As a summary, we get the syllogism table (Table 3). In the table, (t_1) and (t_2) are the truth values of the premises; (F_{ded}), (F_{abd}), and (F_{ind}) representing the truth value calculated by the functions for deduction, abduction, and induction, respectively; f_{ded}, f_{abd}, and f_{ind} are functions gotten by interchange t_1 and t_2 in F_{ded}, F_{abd}, and F_{ind}, respectively.

Different from the functions in the revision rule, the functions F_{ded}, F_{abd}, and F_{ind} are not yet completely derived from the interpretation of the truth value. There are many other functions that can satisfy the constraints listed above. The functions are chosen partially due to their simplicity. In future research, these functions may be refined.

The syllogism of NARS differs from Aristotle's ([Aristotle 89] and [Lukasiewicz 51]) in the following aspects:

1. truth value is no longer binary,
2. deduction, induction and abduction are combined,
3. truth value indicates not only shared instances (extension), but also shared properties (intension).

In the first and third points, it differs from Peirce's syllogism, too ([Peirce 32] and [Feibleman 46]).

A distinctive feature of NARS is the implementation of deduction, induction, and abduction in a unified format, as well as the complete symmetry of induction and abduction. There are other efforts in AI to combine different types of inference, but it seems none of them can lead to such a beautiful relation.

These three rules, working together with the revision rule, are responsible for the forward inferences in NARS-2.2. All the new conclusions/conjectures are generated by the syllogism first, then modified by the revision rule.

$J \setminus Q$	$M \subset P$	$P \subset M$
$S \subset M(t)$	$S \subset P$	$S \subset P$ $P \subset S$
$M \subset S(t)$	$S \subset P$ $P \subset S$	$P \subset S$

Table 4: Backward inference rules

4.3 Backward inference

Due to AIKR, NARS cannot afford the resources to exhaustively try all combinations of promises to derive a desired conclusion. Instead, backward inference rules are provided to make the system work in a goal-directed manner.

The backward inference rules of NARS are determined by the following principles:

1. A judgment "$S \subset P\,[t]$" is an answer of a question Q if and only if Q has the form "$S \subset P$", "$? \subset P$", or "$S \subset ?$".

2. A question Q and a judgment J can derive 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.

In such a way, backward inference is just the "inversion" of forward inference, and its usage is to "wake up" the related judgments to answer the input (environment provided) questions.

Following these principles, we can get a backward inference rules table (Table 4). In the table, P can be either a term or a question mark. Since questions have no truth value, no function is attached with the rules.

5 Behaviors in User Interface

5.1 Accepted tasks

NARS, like many other reasoning systems, needs to carry out (at least) the following two types of basic input tasks, which are named from the environment's point of view:

Telling: As an open system, it needs to accept new knowledge told by the environment. The system can either simply store the new knowledge somewhere in its memory, or do some spontaneous forward inference to get derived knowledge.

Asking: The system can use its available knowledge to answer questions asked by the environment. Answers can be found either by matching the question directly with related knowledge, or by backward-forward inference.

5.2 Working mode

In a full-axiomatic reasoning system, the tasks are processed in a sequential and deterministic way:

waiting for user to input a new task,
get a task,
process that task,
report result,
reset the working memory,
waiting for user to input a new task,
......

The characteristics of such a working mode, as implied by the definition of *computation*, are:

1. There is a unique *initial state* where and only where the system can accept input tasks, that means the tasks are processed in a one-by-one way. If a task arrives when the system is still busy with another task, the new task has to wait. Even if the interruption mechanism is taken into consideration, the picture is fundamentally the same.

2. The system can be referred as a *function* that maps input tasks to output results, that means it always provide the same result for the same task, no matter *when* the task is processed.

3. The resources spent on a task is a constant, which mainly depend on the *complexity* of the involved algorithm and the amount of relevant knowledge, but independent to *when* the task is processed.

4. There are some predetermined *final states* where the system will stop working on a task, no matter whether there are other tasks waiting to be processed.

However, under AIKR, NARS cannot work in such a mode. As a real-time system, tasks can show up at any time, and all have deadlines with them, that means the system cannot use a constant time on each task — the time that can be spent is mostly determined by the environment, not by the system. The system even cannot determine the "resources budget" when the task is accepted, since such a budget will be influenced by unpredictable future events: maybe some new tasks will show up.

As a result, NARS has to work in a *controlled concurrent* mode, which is defined as:

1. The system works simultaneously on many tasks in a time-sharing way, no matter how many processors are there (under AIKR, we cannot assume we can assign a processor to each task and subtask). The system can accept input tasks at any time, and begin to work on them immediately.

2. The processing of each task is divided into many *atomic steps*, which is the scheduling unit of processor time, and should take an approximately constant time to be carried out. If in such a step the system find a good result for a input task, the result will be reported to the environment.

3. Each task will be assigned a *priority*, which is *dynamically adjusted* according to how urgent the task is, whether a good answer has been found, whether it is promising, how many other tasks are there, and so on.

4. The probability that a task in chosen for processing is determined by its relative priority. As a result, different tasks are processed *asynchronously*, that is, at different speeds.

5. Some tasks with low priority may be deleted from the waiting list due to limited space.

5.3 Characteristics

Such a working mode make NARS's behaviors in its interface quite different from other reasoning systems':

(1) The environment will attach an *urgency* value and a *duration* value to each input task. The former indicates the task's priority in time-sharing, the latter indicate the deadline for a result. After the deadline, the environment will no longer need a result for it. To make things simple, both values can be represented in a relative scale, and the system can provide defaults if no specific value is provided.

(2) There is no unique "initial state" where the system is waiting for new tasks, as in Turing Machine and other computation models. Though the system as a whole has a initial state, a task can be accepted at many different states.

(3) In NARS, not only the subtasks of a task are processed in parallel, even different input tasks are processed in this way, too. That means, after a new task is accepted by the system, even with the highest urgency, it is still not guaranteed that the system will spend all its resources on it. Some time may still be used to process previous tasks, and results for different tasks will be reported in an unpredictable order. Generally, all tasks (including subtasks) are competing for the system's resources all the time.

(4) Almost all results are *partial* in the sense that only part of the system's knowledge are involved during the derivation of the result. Under AIKR, the system simply doesn't have the time/space resources to provide *global* result for each task.

(5) NARS is *non-monotonic* in the sense that it can overturn a previously reported result. After a partial result are reported for a task, the task is usually not deleted, but only decrease its priority in the competition. Therefore, it is possible for the system to "change its mind" due to new knowledge or further consideration.

(6) There is no "final state". For some tasks, if their urgency are too low, it is possible for them to be completely ignored. If a partial result is reported for a task, usually neither the system nor a human user can predict whether there will be a better result to be reported later, if more knowledge is took into consideration. It is undecidable whether an answer is "THE" answer for the question, since it depend on events that happen in future, such as whether the system can get new knowledge related to the task or whether more time can be spent on it.

(7) After the deadline of a task is passed, the system will stop reporting results for it. But it doesn't mean the system has stopped working on it and its subtasks. Instead, these tasks now become the system's. When resources are still available, the system will work on them for its own benefit — "Maybe these questions will appear again in the future. Who knows? So I'd better still spend some time on them". As the system become more and more complicated, it will become more and more *autonomous*, that is, the processing of its own tasks will strongly effect the system's behavior.

(8) Usually, the deleting of a task from the system is mainly determined by considerations on resources ("I no longer have time to spend on this trivial/hard task"), rather than by logical decisions ("I have completely solved this problem" or "I'm sure I cannot solve it anyway"). Even after such a deleting, the

(sub)tasks that derived previously from the deleted task may still be active in the system — they are also *autonomous* in the sense they are processed for their own sake, not as *means* to achieve other *ends*.

(9) Since the processing context of every task is constantly changed by the existing of other tasks, the system can no longer be referred as a *function* that deterministically mapping an input task to an output result. Usually, if the same task is given to the system at different times, the processing procedures and results will be more or less different. On the other hand, it is possible for the system to either provide no result or more than one results.

In summary, though the system can still be implemented in a von Neumann digital computer, the *processing of a task* can no longer be described as *computation* ([Kugel 86]).

6 Memory Structure and Inference Control

6.1 Chunk, task and knowledge

From the above discussions, we know that the processing of a task in NARS is divided into atomic steps, and in each step a inference rule is used to derive conclusions from two premises. Here we meet another nice property of term logic, that is, all inferences require that the two promises share a common term. Therefore, it is natural to set up a *chunk* for each term, then distribute all judgments and questions into chunks according to the terms that appear in them as subject and predicate. Now each chunk becomes a independent storage/processing unit, so to make parallel distributed processing inherent to NARS.

Under AIKR, it is impossible for all of the judgments and questions in the system (or even in a chunk) to be candidate of premise in each step of inference. To properly focus the attention of the system, all the judgments and questions are divided into two categories, task and knowledge, according to the following principles:

1. All questions are tasks;
2. All judgments are knowledge;
3. All new knowledge (environment provided or system generated) also have a copy served as a task.

By this division, what we get is:

1. a small set of tasks, which are active, kept for a short time, and closely related to questions and new knowledge; and
2. a huge set of knowledge, which is passive, kept for a long time, and not necessarily related to current questions and new knowledge.

Now, we can think the memory and working space of NARS as a set of chunks, and each chunk consists of a set of tasks and a set of knowledge.

6.2 Atomic step

The atomic step of NARS can be briefly described as the following sequence of operations:

to choose a chunk,
to choose a task from that chunk,
to choose a piece of knowledge from that chunk,
to use the task and knowledge to do inference,
to send the new tasks to corresponding chunks.

With different combination of the task and knowledge, the inference may be one of the following:

- wake up — if the task is a question, and the knowledge happen to be an answer to the question, a copy of the knowledge is generated as a new task;
- backward inference;
- revision; or
- syllogistic inference.

The later three have been introduced earlier. Unlike many other systems, NARS doesn't decide what type of inference is used to process a task when the task is accepted, but works in a *data-driven* way, that is, it is the task and knowledge that happen to be picked up that decide what type of inference will be carried out.

6.3 Resources distribution

As mentioned above, under AIKR it is impossible to process the tasks to their logical end (such as find all possible answers for a question, or find all implied conclusions for a piece of new knowledge). We can neither afford resources to do it, nor can we recognize such a logical end. Therefore, we have to settled for partial solutions, that is, only use part of the system's knowledge to process a task. That doesn't mean any result is equally good. To use the knowledge and resources as efficiently as possible, the system should pay more attention to the most *urgent* and *premising* tasks, and using the more *important* and *relevant* knowledge to process them.

To be concrete, the basic idea is to assign relative priorities to chunks (where they are called *activity*), tasks (where they are called *urgency*), and knowledge (where they are called *importance*), then distribute the system's resources accordingly. NARS-2.2 use a probabilistic algorithm to make the choice. Roughly speaking, if the ratio of priorities of two items (chunk, task or piece of knowledge) is r, then r is also the ratio of their probabilities to be chosen at the current time. After each atomic step, the priorities of the involved chunks, tasks and knowledge will be adjusted according to the results of the step.

There is a set of functions that responsible for the priority calculations.

The *urgency* of an input task is assigned by the environment, and the urgency of a system generated task is calculated from several factors, such as how urgent its parent task is, how important and confident its parent knowledge is, what type of relation is there between it and its parents (deduction is a strong relation, while abduction and induction are weak), and so on. On the other hand, the urgency of a task will decay as more time has been spent on it.

The *activity* of a chunk is positively proportional to the sum of urgencies of the tasks in that chunk.

The *importance* of a piece of knowledge is adjusted after each step of inference where the knowledge is used according to how urgent the generated tasks are, which indicate how useful the knowledge is to the system.

With all these parameters adjusted dynamically, some tasks are processed faster, and some knowledge is more accessible, while the others are relatively forgotten by the system. Because the storage space is limited, chunks, tasks and knowledge with low priorities may also be forgotten permanently.

The control strategy used in NARS shares some properties with the *parallel terraced scan* strategy introduced in [Hofstadter and Mitchell 92] and [Hofstadter and French 92].

6.4 Network interpretation

As another interesting property, NARS can also be interpreted as a *network* ([Wang 92]). We can see each term as a *node*, and each statement as a *directed link* between two terms, and the corresponding truth value as the *weight* of the link. There are *active links* (tasks) and *passive links* (knowledge). *Priorities* are defined among nodes (as activity), active links (as urgency), and passive links (as importance). In each atomic step, an active link interacts with a adjacent passive link to generate new links or to change the weights of existing links, and different types of inference correspond to different combinations of the two links. After each step, related priorities are adjusted.

Such a network is similar to the Slipnet as in [Hofstadter and Mitchell 92] and [Hofstadter and French 92] since they are dynamically adjusted, and it shares many properties with the *subsymbolic* paradigms ([Smolensky 88] and [Holland 86]), such as self-organization, parallel processing, nondeterministicity, distributed representation, and so on.

At the first glimpse, the internal representation seems to be local, since the knowledge "Swans are birds" is stored explicitly in the link that between node *swan* and node *bird*. However, when the system is told about that, the judgment is treated both as a passive link (to be set up between the related nodes) and as an active link (to actively interact with other links), therefore, it will have effects in other nodes and links. On the other hand, when the system is asked "Are swans birds?", it will not only search matching link between node *swan* and node *bird*, but also try to answer the question by inference from available knowledge, and other nodes and links may be involved in the process.

Since in NARS deduction, induction and abduction are closely related, all premises can be rebuilt from the conclusions to some extent. For example, if initially we have judgments J_1:

$$M \subset P \, (t_1),$$

and J_2:

$$S \subset M \, (t_2),$$

we can deduce J_3:

$$S \subset P \, (t_3).$$

On the other hand, if J_1 is no longer accessible in a later time, the system still can get J_4:

$$M \subset P \, (t_4)$$

by induction from J_2 and J_3. J_4's confidence value will be much smaller than J_1's, but it will still keep some of J_1's information.

As a result, the internal representation become *distributed* to certain extent in the sense that

- input task may have non-local effects,
- output result may have non-local sources, and
- local information damage (by forgetting or hardware causes) may be partially recovered from information kept in other part of the system.

When the system has a big enough knowledge base, no task can be actually processed by accessing all of the knowledge. Only certain pieces of knowledge in certain chunks are involved, that means there is knowledge that within the system's memory but cannot be recalled at some situations. On the other hand, the "meaning" of a term to the system, that is, the knowledge associated with it by the system, is fluid and context dependent ([Wang 92]). For example, in the processing of task A, term *swan* is only used in the sense "Swans are birds", while in the processing of task B, it is only used in the sense "Swans are swimmers".

7 Implementation

NARS-2.2 is implemented in Scheme. All source codes occupy about 30K memory. In the following I'll introduce some technical details that not mentioned above.

7.1 Representation and interpretation

To make things simple, in NARS-2.2 we only use the $< f, c >$ form of truth values in the internal representation as well as in the user interface. All other forms of truth value (counting, interval, or verbal) should be translated into this form by human users.

According to the interpretation of the truth value, we make the following conventions to correspond verbal form of truth values provided by users in natural language to NARS's formal representation of truth value:

- Positive judgments without qualifiers (like "Birds are animals"), as well as with universal qualifiers (like "all"), are give a truth value $< 1,\ 0.9 >$, and the corresponding negative judgments are given a truth value $< 0,\ 0.9 >$.
- "Some" is translated into $< 1,\ 0.3 >$, and the corresponding negative judgments are given a truth value $< 0,\ 0.3 >$.
- "Normal" and "usual" are translated to $< 0.9,\ 0.9 >$, and the corresponding negative judgments are given a truth value $< 0.1,\ 0.9 >$.

At the current stage, I don't intend to set up a complete and accurate mapping between verbal truth values and numeral truth values. As long as the above conventions are intuitively reasonable and used consistently, they are good enough for the current purpose.

7.2 Inference rules

The k parameter appearing in the induction rule and the abduction rule is 2. That means, in NARS-2.2, "in the near future" is interpreted as "after the next two pieces of evidence show up".

Before the applying of the revision rule, it is necessary to check whether the two premises are independent to each other. A "postmark" mechanism is used for this purpose. Each piece of input judgment is automatically assigned a unique postmark (an integer) when accepted by the system. In each inference step, the conclusion get a postmark list by interweaving its parents' (the premises') postmark lists, then cut it to a certain length. In NARS-2.2, the maximum length of postmark list is 4. Therefore, if two judgments have a common grandparent, they must have a common postmark. Before revision, two premises' postmark lists are compared. The promises are considered independent to each other if and only if their postmark lists have no common elements. This mechanism is only an approximation, but it is good enough in most cases.

There is a special rule used to recognize results that should be reported to the environment. For each input task, a piece of pseudo-knowledge is inserted into the corresponding chunk(s), and it is used to check possible results and to record the best result that has been reported. When a telling task is matched with the pseudo-knowledge, it will be reported if it is better than the recorded result. The pseudo-knowledge will be forgotten by the system after a period corresponding to that input task's duration (which is specified by the environment). After that, no more result is reported about the input task.

7.3 Memory structure

The memory of NARS-2.2 consists of a *active chunk table* with a maximum capacity of $a_h \times a_w$ and a *dormant chunk list* with a maximum capacity of d_l. Active chunks participate in time-space resources competition, while dormant chunks compete for space only.

In the current version of NARS, the parameter a_h is set to 80, a_w to 2, and d_l to 100. That means the *activity* of a chunk is an integer in $\{1, \cdots, 80\}$, and at each activity level there are most 2 chunks. If a chunk is inserted into a full level of the table, one of the 2 chunks that previously there will be moved to a lower level. Especially, a chunk in level 1 will be moved to the head of the dormant chunk list. If overflow happens to the dormant chunk list, the chunk in the tail will be deleted from the system. The overflow handling (or *forgetting*) mechanism is necessary, since under AIKR a system's storage space is limited.

A chunk can also be accessed by its *name*, that is, the common term shared by all tasks and knowledge in that chunk.

Chunk (active or dormant) consists of a *task table* and a *knowledge table*. The task table's maximum capacity is $u_h \times u_w$. In NARS-2.2 I let $u_h = 20$ and $u_w = 3$. That means the *urgency* of a task is an integer in $\{1, \cdots, 20\}$, and at each urgency level there are most 3 tasks. There is also a overflow handling mechanism, which is similar to the active chunk table with the exception that a task removed from level 1 will be deleted from the system.

The knowledge table is further divided into 4 subtables, and each of them have a maximum capacity of $i_h \times i_w$. In NARS-2.2 I let $i_h = 60$ and $i_w = 2$. That means the *importance* of a piece of knowledge is an integer in $\{1, \cdots, 60\}$, and at each importance level there are most 2 pieces of knowledge. There is also a overflow handling mechanism, which is identical to the task table. For a chunk whose name is S, the four subsets are for knowledge whose statement has the form "$S \subset W$", "$X \subset S$", "$S \subset \{Y\}$", and "$\{Z\} \subset S$", respectively.

Totally, NARS-2.2 can handle no more than $80 \times 2 + 100 = 260$ chunks, with no more than $20 \times 3 = 60$ tasks and $4 \times 60 \times 2 = 480$ pieces of knowledge in each chunk.

7.4 Atomic step

Each atomic step of NARS-2.2, as a time scheduling unit, is consists of the following operations:

choose a chunk: As discussed before, the choice is based on the activities of the chunks.

choose a task from that chunk: As discussed before, the choice is based on the urgencies of the tasks.

choose a subtable of knowledge: The choice, which is made within the same chunk, is based on the type of task and the total importance of knowledge in each subtable. Each task is processed in a chunk in two stages: For new knowledge, the system at first check for possible revisions locally, then do syllogistic inferences. For new questions, the system at first check for possible matching with knowledge, then do backward inference. In syllogistic and backward inferences, deduction has a higher priority than induction and abduction.

choose knowledge from the subtable: To speed up inferences, as well as to provide a inhibitional mechanism among tasks, in each atomic step several (≤ 4 in NARS-2.2) pieces of knowledge is used to process a task.

do inferences: Use the task and each piece of knowledge as premises to get conclusions. Each conclusion is a new task.

adjust the priorities: The the involved chunk, task and knowledge are re-evaluated. The adjusting functions are given by the control strategy of the system.

send the new tasks to corresponding chunks: Each task is then insert into the task table of the related chunk. If it's a "telling" task, a copy of it is also inserted into the corresponding subtable of knowledge. In this way, the related chunks are activated according to the urgency of the tasks that they received, and chunks can cooperate with each other to solve problems.

7.5 Control strategy

Here is, briefly, how the priorities are determined and adjusted in NARS-2.2:

The *activity* of a chunk is proportional to the sum of the urgencies of the tasks in the chunk at current time.

The *urgency* of an input task is determined by the user.

The urgency of a derived task is the scaled product of:

- the urgency of its parent task,
- the importance of its parent knowledge,
- the confidence of its parent knowledge, and
- a inference type parameter, which is 1 for deduction and 2/3 for induction and abduction.

After an atomic step, the task which is processed in the step will decrease its urgency proportionally to the sum of the urgencies of the derived tasks in that atomic step.

The importance of a piece of knowledge is initially proportional to the urgency of the task that bring the knowledge to the chunk. Then, after each time it is used in inference, its importance is adjusted according to the urgency of the derived tasks, that is, whether the knowledge is "useful" in solving the current problem.

7.6 User interface

The user interface of NARS-2.2 provides the following commands for user:

initialization: The memory is reset to empty.

save(file-name): The complete context is saved to the specified file.

return(file-name): The complete context is recovered from the specified file.

run-instruction(instruction): An instruction is either an integer to indicate the number of atomic steps to be carried, or an input task, with its urgency and duration, to be inserted into the corresponding chunk(s).

run-program(program): A program is a sequence of instructions, which will be excused one by one. It will make the test of the system faster and easier.

display utilities: They are used to show the priority distributions within the memory or within a specific chunk for the purpose of debugging and monitoring.

parameters adjustment: There is a set of parameters that specifying aspects of the system, such as the size of a chunk, the decay speed of knowledge, the k parameter that appear in the interpretation, and so on. They can be adjusted by a human designer/user between one run of the system and another run, so the system's "personality" can be tuned to some extent.

8 An Example

Different from expert systems, NARS is a domain-independent reasoning system. It can accept any task, as long as representable in its interface language. The following example has been used to test the system, and the complete experiment results are in the Appendix.

In this example, the system is provided with knowledge about birds, then use the knowledge to answer questions. The instructions are read from a data file.

(1) Background knowledge

After loading the system and initialize its memory, the system is provided with the following knowledge:

Doves are birds.
Normal doves are flyers.
Doves are not swimmers.
Some doves are white.
Some doves are not white.
Swans are birds.
Normal swans are flyers.
Normal swans are swimmers.
Normal swans are white.
Penguins are birds.
Penguins are not flyers.
Normal penguins are swimmers.
Birds are animals.

After a piece of knowledge is inserted, the system is given some time to digest it, that is, to do some spontaneous inferences. We can see that the system can find some inconsistence among the inputs, and some revision results are reported. For example, "$dove \subset white < 1, 0.3 >$" and "$dove \subset white < 0, 0.3 >$" are explicitly inconsistent with each other, so they are summarized into "$dove \subset white < 0.5, 0.462 >$". From the knowledge about *dove* and *swan*, the system guess that normal birds are flyers, but such a guess is implicitly inconsistent with the knowledge about *penguin*. As a result, "*penguin* $\subset$ *flyer* $< 0, 0.9 >$" is slightly modified to "*penguin* $\subset$ *flyer* $< 0.063, 0.906 >$". At the same time, the degree of belief of "*bird* $\subset$ *flyer*" is notably decreased, but since the inductive conclusion is not a input knowledge, its revision is not reported.

(2) Deduction
Now the system is told: "Colomba is a swan." Then, when asked to find a specific instance of "animal", the system answers "Colomba". To get the answer, the system need (at least) to use three pieces of its knowledge to do two steps of deduction. The confidence of the conclusion is lower than any premises', so it is easier to be revised by future evidence.

(3) Induction
When asked "Are birds flyers?", the result's truth value is $< 0.6, 0.548 >$. The answer is gotten by inducing from three pieces of evidence (*dove*, *swan*, and *penguin*) independently, then merging the conclusions together through revision. For the symmetric question "Are flyers birds?", the answer's truth value is more positive but less confident ($< 1.0, 0.421 >$), since only two pieces of positive evidence are observed. Each of the results are reported twice, since the tasks are processed both in chunk *bird* and chunk *flyer*. Generally, results from different chunks are not necessarily to be the same.

(4) Abduction
The system is provided with knowledge "Cigno is white" and "Cigno is a flyer", then is asked "Is Cigno a dove?", "Is Cigno a swan?", "Is Cigno a penguin?", "Is Cigno a bird?", and "Is Cigno a animal?". According to our interpretation, "Is Cigno a X?" actually means "to what extent $\{cigno\}$ inherit X's properties and X inherit $\{cigno\}$'s instances?" These questions are answered by abduction in the sense that the system compare $\{cigno\}$'s properties with properties of *dove*, *swan*, and so on, to see to what extent they are matched. This type of task can also be referred as "analogy", "classification", or "high level perception" in different contexts. From the results, we see that the order of the degree of belief of the results is (decreasingly): *swan*, *bird*, *animal*, *dove*. "Is Cigno a penguin?" doesn't get an answer. As mentioned previously, it is not guaranteed that every question will eventually get an answer from the system. We can also see that the confidence of "*cigno* $\in$ *dove*" is increased when more evidence is processed.

(5) Non-monotonic reasoning
The system is told "Tweety is a bird", then asked "Is Tweety a flyer?". Without further knowledge, the answer is "*tweety* $\in$ *flyer* $< 0.6, 0.493 >$" (with a degree of belief 0.549 — a little more than 0.5), which can be translated into something like "I guess it is". Then the system is told "Tweety is a penguin". With the previous question still active, the system changes its mind, and report something like "Tweety is very unlikely to be a flyer" (formally, "*tweety* $\in$ *flyer* $< 0.063, 0.815 >$", with a degree of belief 0.142). Here we can see why I said earlier that there is no final answer for a question. All results are defaults in the sense that it is always possible for the system to change its mind.

The whole testing takes about 1.5 minute on a SPARC station 1. The results are context sensitive, in the sense if we change the order of the instructions, or change the time interval between the input tasks, the results will be slightly different.

From this experiment we can see that NARS can carry out several types of inference, and they are combined to process tasks. For example, the inductive conclusion "Most birds are flyers" can later be used as a premise in deduction. On the other hand, what type of inference is used to process a task depends on the type of the task and available knowledge.

I also tried to use "The Jets and The Sharks" problem ([McClelland and Rumelhart 88]) to test the previous version of the system, and showed that the system can carry out functions similar to those of a connectionist network, such as retrieval by name (through matching questions to knowledge), retrieval from partial description or noisy description (through abduction), spontaneous generalization (through induction), as well as damage recovery (due to distributed representation) ([Wang 93]).

9 Limitations and Extensions

I'll distinguish three types of limitations for NARS-2.2:

1. limitations that can be removed in the future versions of the system,
2. limitations that may or may not be removed in the future versions of the system, and

3. limitations that cannot be removed in the future versions of the system.

9.1 Short-term limitations

NARS-2.2 is an intelligent reasoning system, in the sense that it is an adaptive system working under AIKR. It is, at the same time, a finite system, a real-time system, an ampliative system, an open system, and a self-organized system. However, this doesn't mean it cannot be made "more intelligent". Actually, according my working definition of intelligence, a system can increase its intelligence by extending its interface language, applying new inference rules, increasing its resources efficiency, self-organizing at a higher level, and so on.

The interface language of NARS-2.2 is very primitive, since only the "$\subset$" and "$\in$" relations between simple terms can be represented. In the next version, I'll try to set up a complete "first order term-oriented language", which will include the following major extensions:

- Define the *similarity* (or *equivalence*) relation "$=$" as the third inheritance relation between two terms. "$S = P$" is identical to "$S \subset P$ and $P \subset S$". Additional inference rules are needed to manage inferences involving the similarity relation.
- A term may be compound, that is, when S and T are terms, $S \cup T$ (*union*), $S \cup T$ (*intersection*), and $S - T$ (*difference*) are valid terms. Additional inference rules are needed to manage inferences involving the composing and decomposing of compound terms. In the new version, the system should be able to automatically generate new terms that can efficiently abstract its knowledge.
- *Relations* will be defined as Cartesian products of terms, and the *image* of a term under a relation will also be defined. New inference rules will be introduced to process these structures. In this way, the system can represent and use knowledge like "John and Mary is a couple" and "Mary is John's wife".

Beside the extensions of the language and the related rules, there will be other improvement in the new version:

- NARS-2.2's control strategy is too simple to deal with complex problems. Some refinments are needed to improve the system's time-space efficiency. The ideas that will lead to the improvement will mainly come from psychological evidence and theories, for example, NARS can learn a lot from the psychological study on memory and attention.
- The system's user interface need to be redesigned to become more user-friendly, and to provide a more vivid picture about how the system works.

Since I already have some comparatively clear ideas about how to overcome these limitations of the current system, they are short-term limitations that only apply to NARS-2.2, but not to future versions of the system. The new version should be completed by the middle of 1994.

9.2 Long-term limitations

Even after the implementation of the new version introduced above, that are still many important and interesting things that NARS cannot do. The following are some of them.

Higher order judgments: It's easy to extend the first order term-oriented language to higher order: we only need to allow a statement or judgment 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 higher order judgments. It seems that all the old rules (those for first order judgments) are still valid, but there should be something specially for the 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 them in planning. But at current time it is still not clear how to extend a term logic to do this.

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 sensory-motor mechanism, its interface language will be greatly extended, therefore will be more intelligent than a system that only have a symbolic language interface. However, such an extension need to use higher order judgments and procedural knowledge, therefore it is more difficult.

Natural language interface: Semantically, NARS-2.2's interface language is already more similar to natural language than many other reasoning systems, since the meaning of a term or a judgment is determined by available knowledge, therefore is fluid and context-sensitive. However, to make NARS handle natural language is still a distant goal.

Meta-level self-organizing: All of NARS-2.2's (or in the near future, NARS-3's) self-organizations are in the domain knowledge level. It can generate new judgments and new terms, and it also can adjust truth values of judgments and priorities of chunks, tasks and knowledge to build and modify its knowledge hierarchy, but it cannot change the system's personality parameters (such the size of a chunk), inference rules (such as how to revise a judgment), or control strategy (such as to change its forgetting rate). These meta-level self-organizations require higher order judgments, procedural knowledge, as well as knowledge about the system itself.

Local axiomazition: Even though I claim that intelligence typically happens under AIKR, it is not contradict with the fact that human beings can locally axiomize a specific domain by assuming sufficient knowledge and resources. In this way, it is possible for the system to use numbers and other mathematical configurations, make counterfactural assumptions, design algorithms, and so on. For such knowledge, we can let their confidence to be 1, that means they are *conventions* made by the system, so will not be directly used to predict future events, and therefore cannot be refused by new evidence. How to coordinate these *analytical knowledge* with the system's other empirical knowledge is still an open problem.

Since I don't have clear ideas about how to extend NARS to do these things yet, they will remain to be NARS's limitations for a long period, and at the same time become my goals in the future years. However, I still have reasons to believe that it is better to solve these problems from a non-axiomatic point of view, rather than from a full-axiomatic or semi-axiomatic point of view, since all these problems are closely related to NARS's working definition of intelligence, that is, to adapt under AIKR. Therefore, what I'm doing now is a necessary step toward these goals.

9.3 Permanent limitations

There are limitations that cannot be removed in the NARS project, since they are fundamentally inconsistent with NARS's working definition of intelligence.

- Since NARS works under AIKR, it is impossible for it to have the properties that only a full-axiomatic system can have, such as *consistency*, *completeness*, and *decidability*. When NARS is used to solve practical problems, it cannot guarantee that its results are all *correct* or *optimal*.

- From a theoretical point of view, NARS is not designed to be an accurate model of human reasoning, but to be a reasoning system that has intelligence (according its working definition of the concept). The system should follow the same *principles* with human mind. However, it is not necessary to have the same internal *structure* and *mechanism* with human brain, since its hardware is fundamentally different. Moreover, since it experience is different from a human being's, it is not necessary (though still possible) to have the same external behaviors with human mind, such as exactly reproduce some psychological data or pass a certain type of Turing test.

- From a practical point of view, NARS is not designed to solve certain domain problems. It is not an expert system or other type of computer application system. It is intelligent, not because it can solve problems that no (or few) people can solve (though it is possible in the future), but because it works in a *human way*. It will make not only human achievements, but also human mistakes.

These limitations are easier to deal with than the previous ones — we can just ignore them. But they are still limitations in the sense that if someone is looking for a computer model for there purposes, then NARS shouldn't be a candidate, since it is designed to achieve other goals.

10 Conclusions

The most important ideas in NARS that distinguish it from other AI systems and cognition models can be summarized as the following:

1. Its *working definition of intelligence*, which stresses the insufficiency of knowledge and resources;

2. Its *term-oriented language*, which represents domain knowledge in the form of inheritance relations between terms;

3. Its measurement and interpretation of *truth value* of a judgment, which covers several types of uncertainty;

4. Its *inference rules*, in which deductive, inductive, abductive, revisive, and backward inferences are combined in a uniform way;

5. Its *controlled concurrence* working mode, which allows the system to work in real time;

6. Its *chunk-based memory structure*, which provides a fluid, dynamic, and context-dependent way to organize knowledge.

All these ideas are closely related with each other, so it is almost impossible to understand or use some of them without understanding or using the others.

From NARS-2.2's practice, I hope to show that it is possible to establish a fully formalized reasoning system under the assumption that its knowledge and resources are usually insufficient to process its tasks. Though relatively simple, NARS-2.2 is indeed a finite system, a real-time system, an ampliative system, an open system, and a self-organized system. It doesn't claim to always find correct or optimum answers for questions, but its behavior is similar to the common sense reasoning of human mind in many aspects.

The study of artificial intelligence, with all its heritage from mathematical logic and computation theory, is still closely bound to the assumption of sufficient knowledge and resource. Given the properties of most AI domains, AIKR is often treated by AI researchers as restrictions on the *implementation* of a theory of intelligence, rather than on the theory itself. As a result, many "intelligent systems" can solve *practical problems*, but can only do so under *impractical conditions*, that is, with well-organized domain knowledge provided by knowledge engineers and enough resource that can be used for brute-forced search/calculation. What we call *intelligent* behaviors in human beings seldom happen in such a situation.

However, what should be blamed are not the ideas of *symbolization*, *formalization*, and *logic* (as suggested by [Dreyfus 92], [Lucas 61], [Searle 84], and others), but the ideas of *axiomazition* and *computation*. It is true that at a low level, NARS is still implemented through a *axiomatic computational system*, but itself cannot be reduced into such a system at the level where it communicates with its environment. Symbolic systems, especially logical reasoning systems, still have advantages in reproducing high-levels intelligent behaviors.

Acknowledgement

I thank David Leake and Steve Larson for their helpful comments on the draft of the paper.

References

[1] Aristotle (1989). *Prior Analytics.* Hackett Publishing Company.

[2] Bhatnagar R. and Kanal L. (1986). Handling uncertain information. In Kanal, N. and Lemmer, J. (eds.) *Uncertainty in Artificial Intelligence.* North-Holland.

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

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

[5] Cherniak, C. (1986). *Minimal Rationality.* The MIT Press.

[6] Dreyfus, H. (1992). *What Computers still Can't Do.* The MIT Press.

[7] Falkenhainer, B. (1990). A unified approach to explanation and theory formation. In Shavlik, J. and Dietterich, T. (eds.) *Readings in Machine Learning.* Morgan Kaufmann Publishers.

[8] Feibleman, J. (1946). *An Introduction to Peirce's Philosophy.* Harper and Brothers Publishers.

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

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

[11] Hofstadter, D. and French, R. (1992). Probing the emergent behavior of Tabletop, an architecture uniting high-level perception with analogy-making. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society.*

[12] Hofstadter, D. and Mitchell, M. (1992). The Copycat project: a model of mental fluidity and analogy making. To appear in Holyoak, K. and Barnden, J. (eds.) *Connectionist and Neural Computation Theory, Volume 2: Analogical Connections.* Ablex Publishing Corporation.

[13] Holland, J. (1986). Escaping brittleness. In Michalski, R. *et al.* (eds.) *Machine Learning II.* Morgan Kaufmann Publishers.

[14] Kugel, P. (1986). Thinking may be more than computing. *Cognition 22.*

[15] Lucas, J. (1961). Minds, machines, and Gödel. *Philosophy 36.*

[16] Łukasiewicz, J. (1951). *Aristotle's Syllogistic: From the Standpoint of Modern Formal Logic.* Oxford University Press.

[17] McClelland, J. and Rumelhart, D. (1988). *Explorations in Parallel Distributed Processing.* MIT Press.

[18] Michalski, R. (1983). A theory and methodology of inductive learning. In Michalski, R. *et al.* (eds.) *Machine Learning.* Morgan Kaufmann Publishers.

[19] Michalski, R. and Kodratoff, Y. (1990). Research in machine learning. In Kodratoff, Y. and Michalski, R. (eds.) *Machine Learning III.* Morgan Kaufmann Publishers.

[20] Paaß, G. (1991). Second order probabilities for uncertain and conflicting evidence. In Bonissone, P. *et al.* (eds.), *Uncertainty in Artificial Intelligence 6.* North-Holland.

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

[22] Peirce, C. (1932). *Collected Papers of Charles Sanders Peirce* (Vol. 2). Harvard University Press.

[23] Reiter, R. (1987). Nonmonotonic reasoning. *Annual review of computer science 2.*

[24] Russell S. and Wefald, E. (1991). *Do the Right Thing.* The MIT Press.

[25] Searle, J. (1984). *Minds, Brains and Science.* Harvard University Press.

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

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

[28] Spies, M. (1989). *Syllogistic Inference under Uncertainty.* Psychologie Verlags Union.

[29] Tversky, A, and Kahneman, D. (1974). Judgment under uncertainty. *Science 185.*

[30] Wang, P. (1986). A Reasoning System That Can Deal with Uncertainty. Master Thesis, Peking University.

[31] Wang, P. and Hsu, C. (1987). A discovery-oriented logic model. *Proceedings of the Second International Conference on Computers and Applications* (Beijing), IEEE press.

[32] Wang, P. (1992). First Ladies and Fluid Logics. Technical Report (No. 62) of the Center for Research on Concepts and Cognition, Indiana University.

[33] Wang, P. (1993). Non-Axiomatic Logic (Version 2.1). Technical Report (No. 71) of the Center for Research on Concepts and Cognition, Indiana University.

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

[35] Zadeh, L. (1985). Syllogistic reasoning in fuzzy logic and its application to usuality and reasoning with dispositions. *IEEE Transactions, SMC-15.*

Appendix

Notes:

1. The lines lead by ">>>" are instructions, and the lines lead by "<<<" are reported results.
2. The character "%" is used to indicate "$\subset$", and the character "@" is used to indicate "$\in$".

```
Chez Scheme Version 4.0a
Copyright (c) 1991 Cadence Research Systems

> (load "nars")

> (set-bg)

  >>> NEW TASK [urgency=15, duration=3]: dove % bird <1, 0.9>

  >>> 4 ATOMIC STEPS

  >>> NEW TASK [urgency=15, duration=3]: dove % flyer <0.9, 0.9>

  >>> 10 ATOMIC STEPS

  >>> NEW TASK [urgency=15, duration=3]: dove % swimmer <0, 0.9>

  >>> 10 ATOMIC STEPS

  >>> NEW TASK [urgency=15, duration=3]: dove % white <1, 0.3>

  >>> 10 ATOMIC STEPS

  >>> NEW TASK [urgency=15, duration=3]: dove % white <0, 0.3>

  >>> 40 ATOMIC STEPS

      <<< RESULT: dove % white <0.5, 0.462>    d = 0.5

      <<< RESULT: dove % white <0.5, 0.462>    d = 0.5

      <<< RESULT: dove % white <0.5, 0.462>    d = 0.5

      <<< RESULT: dove % white <0.5, 0.462>    d = 0.5

  >>> NEW TASK [urgency=15, duration=3]: swan % bird <1, 0.9>

  >>> 20 ATOMIC STEPS

  >>> NEW TASK [urgency=15, duration=3]: swan % flyer <0.9, 0.9>

  >>> 20 ATOMIC STEPS
  >>> NEW TASK [urgency=15, duration=3]: swan % swimmer <0.9, 0.9>

  >>> 20 ATOMIC STEPS

  >>> NEW TASK [urgency=15, duration=3]: swan % white <0.9, 0.9>

  >>> 100 ATOMIC STEPS

  >>> NEW TASK [urgency=15, duration=3]: penguin % bird <1, 0.9>

  >>> 40 ATOMIC STEPS

  >>> NEW TASK [urgency=15, duration=3]: penguin % flyer <0, 0.9>

  >>> 40 ATOMIC STEPS

      <<< RESULT: penguin % flyer <0.063, 0.906>    d = 0.104

  >>> NEW TASK [urgency=15, duration=3]: penguin % swimmer <0.9, 0.9>

  >>> 100 ATOMIC STEPS

  >>> NEW TASK [urgency=15, duration=3]: bird % animal <1, 0.9>

  >>> 300 ATOMIC STEPS

> (ded)

  >>> NEW TASK [urgency=10, duration=3]: colomba @ swan <1, 0.9>

  >>> 200 ATOMIC STEPS

  >>> NEW TASK [urgency=10, duration=3]: ? @ animal

  >>> 300 ATOMIC STEPS

      <<< RESULT: colomba @ animal <1.0, 0.729>    d = 0.864

> (ind)

  >>> NEW TASK [urgency=10, duration=3]: bird % flyer

  >>> 200 ATOMIC STEPS

      <<< RESULT: bird % flyer <0.6, 0.548>    d = 0.555

      <<< RESULT: bird % flyer <0.6, 0.548>    d = 0.555

  >>> NEW TASK [urgency=10, duration=3]: flyer % bird

  >>> 250 ATOMIC STEPS
```

```
        <<< RESULT: flyer % bird <1.0, 0.421>    d = 0.71

        <<< RESULT: flyer % bird <1.0, 0.421>    d = 0.71

> (abd)

  >>> NEW TASK [urgency=20, duration=3]: cigno @ white <1, 0.9>

  >>> 200 ATOMIC STEPS

  >>> NEW TASK [urgency=20, duration=3]: cigno @ flyer <1, 0.9>

  >>> 300 ATOMIC STEPS

  >>> NEW TASK [urgency=15, duration=5]: cigno @ dove

  >>> 250 ATOMIC STEPS

        <<< RESULT: cigno @ dove <1, 0.132>    d = 0.566

        <<< RESULT: cigno @ dove <1, 0.132>    d = 0.566

        <<< RESULT: cigno @ dove <1, 0.267>    d = 0.634

  >>> NEW TASK [urgency=15, duration=5]: cigno @ swan

  >>> 250 ATOMIC STEPS

        <<< RESULT: cigno @ swan <1.0, 0.415>    d = 0.708

        <<< RESULT: cigno @ swan <1.0, 0.415>    d = 0.708

  >>> NEW TASK [urgency=15, duration=5]: cigno @ penguin

  >>> 250 ATOMIC STEPS

  >>> NEW TASK [urgency=15, duration=5]: cigno @ bird

  >>> 250 ATOMIC STEPS

        <<< RESULT: cigno @ bird <1.0, 0.379>    d = 0.69

        <<< RESULT: cigno @ bird <1.0, 0.379>    d = 0.69

  >>> NEW TASK [urgency=15, duration=5]: cigno @ animal

  >>> 400 ATOMIC STEPS

        <<< RESULT: cigno @ animal <1.0, 0.341>    d = 0.67
```

```
    <<< RESULT: cigno @ animal <1.0, 0.341>    d = 0.67

> (nm)

  >>> NEW TASK [urgency=20, duration=3]: tweety @ bird <1, 0.9>

  >>> 200 ATOMIC STEPS

  >>> NEW TASK [urgency=20, duration=50]: tweety @ flyer

  >>> 200 ATOMIC STEPS

    <<< RESULT: tweety @ flyer <0.6, 0.493>    d = 0.549

    <<< RESULT: tweety @ flyer <0.6, 0.493>    d = 0.549

  >>> NEW TASK [urgency=20, duration=3]: tweety @ penguin <1, 0.9>

  >>> 100 ATOMIC STEPS

    <<< RESULT: tweety @ penguin <1.0, 0.904>    d = 0.952

    <<< RESULT: tweety @ flyer <0.063, 0.819>    d = 0.142

    <<< RESULT: tweety @ penguin <1.0, 0.904>    d = 0.952

>
```