# Toward a Unified Artificial Intelligence

**Pei Wang**
Department of Computer and Information Sciences
Temple University
pei.wang@temple.edu
http://www.cis.temple.edu/∼pwang/

**Abstract**

To integrate existing AI techniques into a consistent system, an intelligent core is needed, which is general and flexible, and can use the other techniques as tools to solve concrete problems. Such a system, NARS, is introduced. It is a general-purpose reasoning system developed to be adaptive and capable of working with insufficient knowledge and resources. Compared to traditional reasoning system, NARS is different in all major components (language, semantics, inference rules, memory structure, and control mechanism).

## Intelligence as a whole

Artificial intelligence started as an attempt to build a general-purpose thinking machine with human-level intelligence. In the past decades, there were projects aimed at algorithms and architectures capturing the essence of intelligence, such as General Problem Solver (Newell and Simon, 1976) and The Fifth Generation Computer Project (Feigenbaum and McCorduck, 1983). After initial excitement, however, all these projects failed to meet the goal, though they contributed to the AI research here or there.

The lesson many people learned from this history is that there is no such a thing called "intelligence", and it is just a convenient way to address a collection of cognitive capacities and functions of the human mind. Consequently, the majority of the AI community have turned to various concrete problems. Many achievements have been made on these subfields, but are we really approaching the "thinking machine" dream? It is true that a complicated problem has to be cut into pieces to be solved one by one, but if everyone cuts the problem in his/her own way, and only works in a small piece obtained in this manner, we have no reason to believe that the solutions can later be put together to form a solution to the original problem (Brooks, 1991).

People who still associate themselves to the original AI dream find the current situation disappointing. As Minsky said: (Stork, 1997)

> The bottom line is that we really haven't progressed too far toward a truly intelligent machine. We have collections of dumb specialists in small domains; the true majesty of general intelligence still awaits our attack. We have got to get back to the deepest questions of AI and general intelligence and quit wasting time on little projects that don't contribute to the main goal.

Piaget said: (Piaget, 1963)

> Intelligence in action is, in effect, irreducible to everything that is not itself and, moreover, it appears as a total system of which one cannot conceive one part without bringing in all of it.

This opinion does not deny that intelligence includes many distinguishable functions carried out by distinct mechanisms, but it stresses the close relations among the functions and processes, which produce intelligence as a whole.

Technically, it is not difficult to build hybrid systems using multiple AI techniques, by letting several modules (each is based on a different technique) cooperate in the system's activities. However, there are deep issues in this approach. Since different AI techniques are usually developed according to very different theoretical assumptions, there is no guarantee that they can correctly work together. It is easy to pass data from one module to another one, but if the two modules interpret the data differently, the system's integrity can be damaged. Furthermore, if different techniques are applicable to the same type of problems, the system needs a mechanism to decide which technique to use for a new problem, or how to reach a consensus from the results obtained from different techniques.

If the existing domain-specific AI techniques are seen as *tools*, each of which is designed to solve a special problem, then to get a general-purpose intelligent system, it is not enough to put these tools into a toolbox. What we need here is a *hand*. To build an integrated system that is self-consistent, it is crucial to build the system around a general and flexible *core*, as the hand that uses the tools coming in different forms and shapes.

The situation here is also similar to the programs in a computer system, where one program, the operating system, occupies a special position. While the other programs are developed in various ways to solve specific problems outside, an operating system is consistently developed to solve the problem of the system itself, by managing the processes and the resources of the system. What we need now is like an "intelligent operating system" that knows how to run the various AI programs.

It is important to understand that such an "intelligent core" should be designed and evaluated in a way that is fundamentally different from that of the "intelligent tools", because it faces a different problem. An operating system usually does not solve any problem that are solved by the application programs, and a hand is usually not as good as a specially designed tool in solving a specific problem. A system with "general intelligence" does not necessary work better than a non-intelligent one on a concrete problem. Actually the situation is usually the opposite: for any *given* problem, it is always possible to design a tool that works better than our hands. However, with their generality, flexibility, and efficiency, our hands are more valuable than any tools.

If that is the case, then what is the standard of a good intelligent core? It should had the following properties:

- It should be based on a theory that is consistent with the research results of artificial intelligence, psychology, philosophy, linguistics, neuroscience, and so on.
- It should use a technique that is general enough to cover many cognitive facilities, and can be efficiently implemented in existing hardware and software.
- It should be able to use various kinds of tools that are not developed as parts of the system.

In the following, such a system is introduced.

## Theoretical foundation

The system to be discussed in this paper is NARS (Non-Axiomatic Reasoning System). This system is designed according to the belief that *intelligence is the capacity of a system to adapt to its environment while operating with insufficient knowledge and resources* (Wang, 1995).

NARS communicates with its environment in a formal language. The stream of input sentences, representing tasks the system needs to carry out, is the system's *experience*. The stream of output sentences, representing results of the task-processing activity, is the system's *behavior*. The system works by carrying out inference activities. In each inference step, a formal rule is applied to derive conclusions from premises. The memory and control mechanism manages the resources of the system, by distributing the available time-space resources among inference activities.

To *adapt* means that the system learns from its experiences, that is, when processing the tasks, the system behaves in such a way that as if its future experience will be similar to the past experience.

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

**finite:** The system has a constant information-processing capacity.

**real-time:** All tasks have time requirements attached.

**open:** No constraint is put on the content of a task that the system may be given, as long as it is expressible in the formal language.

Psychologists Medin and Ross told us, "Much of intelligent behavior can be understood in terms of strategies for coping with too little information and too many possibilities" (Medin and Ross, 1992). Such an idea is not too novel to AI. Actually, several subfields in AI directly deal with "too little information and too many possibilities", like in heuristic search, reasoning under uncertainty, and machine learning. However, each of the previous approaches usually focuses only on one issue, while NARS is an attempt to address all of these issues. "Insufficient knowledge and resources", defined as above, is a more severe restriction than similar ones like "bounded rationality", "incomplete and uncertain knowledge", and "limited resources".

The framework of a reasoning system is chosen for this project, mainly because of the following reasons:

- It is a general-purpose system. Working in such a framework keeps us from being bothered by domain-specific properties, and also prevents us from cheating by using domain-specific tricks.
- It uses a rich formal language, especially compared to the "language" used in multiple-dimensional space, where a huge number of dimensions are needed to represent a moderately complicated situation.
- Since the activities of a reasoning system consists of individual inference steps, it allows more flexibility, especially compared to the algorithm-governed processes, where the linkage from one step to the next is fixed, and if a process stops in the middle, no valid result can be got.
- Compared with cognitive activities like low-level perception and motor control, reasoning is at a more abstract level, and is one of the cognitive skills that collectively make human beings so qualitatively different from other animals.
- As will be displayed by this paper, the notion of "reasoning" can be extended to cover many cognitive functions, including learning, searching, categorizing, planning, decision making, and so on.

Limited by the paper length, NARS is only briefly described here. For related publications and a prototype of the system (a Java applet), please visit the author's homepage.

## The core logic

The core logic of NARS has been described in detail in (Wang, 1994; Wang, 1995).

NARS uses a categorical language that is based on an *inheritance* relation, "$\rightarrow$". The relation, in its ideal form, is a reflexive and transitive binary relation defined between *terms*, where a term can be thought as the name of a concept. For example, "$raven \rightarrow bird$" is an inheritance statement with "$raven$" as *subject term* and "$bird$" as *predicate term*. Intuitively, it says that the subject is a *specialization* of the predicate, and the predicate is a *generalization* of the subject. This statement roughly corresponds to the English sentence "Raven is a kind of bird".

Based on the inheritance relation, the *extension* and *intension* of a term are defined as the set of terms that are its specializations and generalizations, respectively. That is, for a given term $T$, its extension $T^E$ is the set $\{x \mid x \rightarrow T\}$, and its intension $T^I$ is the set $\{x \mid T \rightarrow x\}$.

Given the reflexivity and transitivity of the inheritance relation, it can be proved that for any terms $S$ and $P$, "$S \rightarrow P$" is true if and only if $S^E$ is included in $P^E$, and $P^I$ is included in $S^I$. Therefore, "There is an inheritance relation from $S$ to $P$" is equivalent to "$P$ inherits the extension of $S$, and $S$ inherits the intension of $P$".

When considering "imperfect" inheritance statements, the above result naturally gives us the definition of (positive and negative) evidence. For a given statement "$S \rightarrow P$", if a term $M$ in both $S^E$ and $P^E$, or in both $P^I$ and $S^I$, then it is a piece of positive evidence for the statement, because as far as $M$ is concerned, the stated inheritance is true; if $M$ in $S^E$ but not in $P^E$, or in $P^I$ but not in $S^I$, then it is a piece of negative evidence for the statement, because as far as $M$ is concerned, the stated inheritance is false; if $M$ is neither in $S^E$ nor in $P^I$, it is not evidence for the statement, and whether it is also in $P^E$ or $S^I$ does not matter.

Let us use $w^+$, $w^-$, and $w$ for the amount of positive, negative, and total evidence, respectively, then we have

$$\begin{aligned} w^+ &= |S^E \cap P^E| + |P^I \cap S^I| \\ w^- &= |S^E - P^E| + |P^I - S^I| \\ w &= w^+ + w^- \\ &= |S^E| + |P^I| \end{aligned}$$

Finally, the truth value of a statement is defined as a pair of numbers $<f, c>$. Here $f$ is called the *frequency* of the statement, and $f = w^+/w$. The second component $c$ is called the *confidence* of the statement, and $c = w/(w+k)$, where $k$ is a system parameter with 1 as the default value. For a more detailed discussion on truth value and its relation with probability, see (Wang, 2001b).

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

This new semantics explicitly defines meaning and truth value in a language used by a system in terms of the experience of the system, and is more suitable for an adaptive system. NARS does not use model-theoretic semantics, because under the assumption of insufficient knowledge, the system cannot be designed according to the notion of a "model", as a consistent and complete description of the environment.

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

NARS uses syllogistic inference rules. A typical syllogistic rule takes two judgments sharing a common term as premises, and derives a conclusion, which is a judgment between the two unshared terms. For inference among inheritance judgments, there are three possible combinations if the two premises share exactly one term:

$$\begin{aligned} &\{M \rightarrow P <f_1, c_1>,\ S \rightarrow M <f_2, c_2>\} \vdash S \rightarrow P <F_{ded}> \\ &\{M \rightarrow P <f_1, c_1>,\ M \rightarrow S <f_2, c_2>\} \vdash S \rightarrow P <F_{ind}> \\ &\{P \rightarrow M <f_1, c_1>,\ S \rightarrow M <f_2, c_2>\} \vdash S \rightarrow P <F_{abd}> \end{aligned}$$

The three rules above correspond to *deduction*, *induction*, and *abduction*, respectively, as indicated by the names of the truth-value functions. In each of these rules, the two premises come with truth values $<f_1, c_1>$ and $<f_2, c_2>$, and the truth value of the conclusion, $<f, c>$, is a function of them — according to the experience-grounded semantics, the truth value of the conclusion is evaluated with respect to the evidence provided by the premises.

These truth-value functions are designed in the following procedure:

1. Treat all relevant variables as binary variables taking 0 or 1 values, and determine what values the conclusion should have for each combination of premises, according to the semantics.

2. Represent the variables of conclusion as Boolean functions of those of the premises, satisfying the above conditions.

3. Extend the Boolean operators into real number functions defined on [0, 1] in the following way:

$$\begin{aligned} not(x) &= 1 - x \\ and(x_1, ..., x_n) &= x_1 * ... * x_n \\ or(x_1, ..., x_n) &= 1 - (1 - x_1) * ... * (1 - x_n) \end{aligned}$$

4. Use the extended operators, plus the relationship between truth value and amount of evidence, to rewrite the functions as among truth values (if necessary).

For the above rules, the resulting functions are:

$$\begin{aligned} F_{ded}: &\quad f = f_1 f_2 \quad c = f_1 f_2 c_1 c_2 \\ F_{ind}: &\quad f = f_1 \quad\ \ c = f_2 c_1 c_2 / (f_2 c_1 c_2 + k) \\ F_{abd}: &\quad f = f_2 \quad\ \ c = f_1 c_1 c_2 / (f_1 c_1 c_2 + k) \end{aligned}$$

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

$$\{S \rightarrow P <f_1, c_1>,\ S \rightarrow P <f_2, c_2>\} \vdash S \rightarrow P <F_{rev}>$$

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

The above function is derived from the additivity of the amount of evidence and the relation between truth value and amount of evidence.

The revision rule can be used to merge less confident conclusions, so as to get more confident conclusions. In this way, patterns repeatedly appear in the experience can be recognized and learned.

## Compound terms

In the history of logic, there are the *term logic* tradition and the *predicate logic* tradition (Bocheński, 1970; Englebretsen, 1996). The former uses *categorical* sentences (with a subject term and a predicate term) and *syllogistic* rule (where premises must have a shared term), exemplified by Aristotle's Syllogistic (Aristotle, 1989). The latter uses *functional* sentences (with a predicate and an argument list) and *truth-functional* rules (where only the truth values of the premises matter), exemplified by first-order predicate logic (Frege, 1970; Whitehead and Russell, 1910).

From the previous description, we can see that NARS uses a kind of term logic. To address the traditional criticism on the poor expressive power of term logic, in NARS the logic allows "compound term", i.e., terms built by term operators from other terms.

First, two kinds of *set* are defined. If $t_1, \cdots, t_n$ ($n \geq 1$) are different terms, an *extensional set* $\{t_1, \cdots, t_n\}$ is a compound term with $t_1, \cdots, t_n$ *as elements*, and an *intensional set* $[t_1, \cdots, t_n]$ is a compound term with $t_1, \cdots, t_n$ as *attributes*.

Three variants of the inheritance relation are defined as the following:

- The *similarity* relation "$\leftrightarrow$" is symmetric inheritance, i.e., "$S \leftrightarrow P$" if and only if "$S \rightarrow P$" and "$P \rightarrow S$".
- The *instance* relation "$\circ\!\!\rightarrow$" is equivalent to an inheritance relation where the subject term is an extensional set with a single element, i.e., "$S \circ\!\!\rightarrow P$" if and only if "$\{S\} \rightarrow P$".
- The *property* relation "$\rightarrow\!\!\circ$" is equivalent to an inheritance relation where the predicate term is an intensional set with a single attribute, i.e., "$S \rightarrow\!\!\circ P$" if and only if "$S \rightarrow [P]$".

If $T_1$ and $T_2$ are different terms, no matter whether they are sets or not, the following compound terms can be defined with them as components:

- $(T_1 \cap T_2)$ is their *extensional intersection*, with extension $T_1^E \cap T_2^E$ and intension $T_1^I \cup T_2^I$.
- $(T_1 \cup T_2)$ is their *intentional intersection*, with intension $T_1^I \cap T_2^I$ and extension $T_1^E \cup T_2^E$.
- $(T_1 - T_2)$ is their *extensional difference*, with extension $T_1^E - T_2^E$ and intension $T_1^I$.
- $(T_1 \ominus T_2)$ is their *intentional difference*, with intension $T_1^I - T_2^I$ and extension $T_1^E$.

The first two operators can be extended to apply on more than two components.

In NARS, only the inheritance relation and its variants are defined as logic constants that are directly recognized by the inference rules. All other relations are converted into inheritance relations with compound terms. For example, a relation $R$ among three terms $T_1$, $T_2$, and $T_3$ can be equivalently rewritten as one of the following inheritance statements:

- $(\times\ T_1\ T_2\ T_3) \rightarrow R$
- $T_1 \rightarrow (\perp\ R\ \diamond\ T_2\ T_3)$
- $T_2 \rightarrow (\perp\ R\ T_1\ \diamond\ T_3)$
- $T_3 \rightarrow (\perp\ R\ T_1\ T_2\ \diamond)$

For each type of statements introduced above, its truth value is defined similarly to that of the inheritance statement. All the inference rules defined in the core logic can be used on statements with compound terms, and there are additional rules to handle the structure of the compounds.

## Higher-order inference

The inference discussed so far is "first-order", in the sense that a statement represents a relation between two terms, while a term cannot be a statement. If a statement can be used as a term, the system will have "higher-order" statements, that is, "statements about statements". The inference on these statements are higher-order inference.

For example, "Bird is a kind of animal" is represented by statement "$bird \rightarrow animal$", and "Tom knows that bird is a kind of animal" is represented by statement "$(bird \rightarrow animal) \circ\!\!\rightarrow (\perp\ know\ \{Tom\}\ \diamond)$", where the subject is a "higher-order term", i.e., a statement.

Compound higher-order terms are also defined: if $T_1$ and $T_2$ are different higher-order terms, so do their *negations* ($\neg T_1$ and $\neg T_2$), *disjunction* ($T_1 \vee T_2$), and *conjunction* ($T_1 \wedge T_2$).

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

- *implication*, "$\Rightarrow$", which intuitively corresponds to "if-then";
- *equivalence*, "$\Leftrightarrow$", which intuitively corresponds to "if-and-only-if".

Higher-order inference in NARS is defined as partially isomorphic to first-order inference. The corresponding notions are listed in the same row of the following table:

| **first-order** | **higher-order** |
|---|---|
| inheritance | implication |
| similarity | equivalence |
| subject | antecedent |
| predicate | consequent |
| extension | sufficient condition |
| intension | necessary condition |
| extensional intersection | conjunction |
| intensional intersection | disjunction |

According to this isomorphism, many first-order inference rules get their higher-order counterparts. For example, NARS has the following rules for (higher-order) deduction, induction, and abduction, respectively:

$$\{M \Rightarrow P < f_1, c_1 >,\ S \Rightarrow M < f_2, c_2 >\} \vdash S \Rightarrow P < F_{ded} >$$
$$\{M \Rightarrow P < f_1, c_1 >,\ M \Rightarrow S < f_2, c_2 >\} \vdash S \Rightarrow P < F_{ind} >$$
$$\{P \Rightarrow M < f_1, c_1 >,\ S \Rightarrow M < f_2, c_2 >\} \vdash S \Rightarrow P < F_{abd} >$$

These rules use the same truth-value functions as defined in first-order inference, though their meanings are different.

There are certain notions in higher-order inference that have no counterpart in first order inference. For instance, by treating a judgment "$S < f, c >$" as indicating that the statement $S$ is implied by the (implicitly represented) available evidence, we get another set of rules (see (Wang, 2001a) for details):

$$\begin{array}{rr} \{M \Rightarrow P <f_1, c_1>,\ M <f_2, c_2>\} \vdash & P <F_{ded}> \\ \{P <f_1, c_1>,\ S <f_2, c_2>\} \vdash & S \Rightarrow P <F_{ind}> \\ \{P \Rightarrow M <f_1, c_1>,\ M <f_2, c_2>\} \vdash & P <F_{abd}> \end{array}$$

In NARS, an "*event*" is defined as a statement whose truth value holds in a certain period of time. As a result, its truth value is time dependent, and the system can describe its temporal relations with other events.

In NAL, time is represented indirectly, through events and their temporal relations. Intuitively, an event happens in a time interval, and temporal inference rules can be defined on these intervals (Allen, 1984). However, in NARS each event is represented by a term, whose corresponding time interval is not necessarily specified. In this way, NARS assumes less information. When the duration of an event is irrelevant or unknown, it can be treated as a point in the stream of time.

The simplest temporal order between two events $E_1$ and $E_2$ can be one of the following three cases: (1) $E_1$ happens before $E_2$, (2) $E_1$ happens after $E_2$, and (3) $E_1$ and $E_2$ happen at the same time. Obviously, the first two cases correspond to the same temporal relation. Therefore, the primitive temporal relations in NARS are "*before-after*" (which is irreflexive, antisymmetric, and transitive) and "*at-the-same-time*" (which is reflexive, symmetric, and transitive). They correspond to the "*before*" and "*equal*" discussed in (Allen, 1984), respectively.

Since "$E_1$ happens before $E_2$" and "$E_1$ and $E_2$ happen at the same time" both assumes "$E_1$ and $E_2$ happen (at some time)", they are treated as "$E_1 \wedge E_2$" plus temporal information. Therefore, we can treat the two temporal relations as variants of the statement operator "*conjunction*" ("$\wedge$") — "*sequential conjunction*" (",") and "*parallel conjunction*" (";"). Consequently, "$(E_1, E_2)$" means "$E_1$ happens before $E_2$", and "$(E_1; E_2)$" means "$E_1$ and $E_2$ happen at the same time". Obviously, "$(E_2; E_1)$" is the same as "$(E_1; E_2)$", but "$(E_1, E_2)$" and "$(E_2, E_1)$" are usually different. As before, these operators can take more than two arguments. These two operators allow NARS to represent complicated events by dividing them into sub-events recursively, them specifying temporal relations among them.

Similarly, there are the temporal variants of *implication* and *equivalence*. For an implication statement "$S \Rightarrow T$" between events $S$ and $T$, three different temporal relations can be distinguished:

1. If $S$ happens before $T$, the statement is called "predictive implication", and is rewritten as "$S \,/\!\!\Rightarrow T$", where $S$ is called a *sufficient precondition* of $T$, and $T$ a *necessary postcondition* of $S$.

2. If $S$ happens after $T$, the statement is called "retrospective implication", and is rewritten as "$S \backslash\!\!\Rightarrow T$", where $S$ is called a *sufficient postcondition* of $T$, and $T$ a *necessary precondition* of $S$.

3. If $S$ happens at the same time as $T$, the statement is called "concurrent implication", and is rewritten as "$S \,|\!\!\Rightarrow T$", where $S$ is called a *sufficient co-condition* of $T$, and $T$ a *necessary co-condition* of $S$.

The same can be done for the "equivalence" relation.

The rules for temporal inference are variants of the rules defined previously. The only additional capacity of these rules is to keep the available temporal information. Since the logical factor and the temporal factor are independent of each other in the rules, these variants can be obtained by processing the two factors separately, then combining then in the conclusion.

In NARS, "*operation*" is a special kind of event, which can be carried out by the system itself. Therefore it is system dependent — the operations of a system will be observed as events by other systems.

Statement "$(\times\ \{A\}\ \{B\}\ \{C\}) \rightarrow R$" intuitively corresponds to "There is a relation $R$ among (individuals) $A$, $B$, and $C$". If $R$ is an event, it becomes "An event $R$ happens among $A$, $B$, and $C$". If $R$ is an operation, it becomes "To carry out $R$ among $A$, $B$, and $C$". We can also say that an operation is a statement under procedural interpretation, as in logic programming.

All the inference rules defined on statements in general can be applied to events; all the inference rules defined on events in general can be applied to operations.

## Task and belief

Every sentence in NARS introduced so far is a judgment, that is, a statement with a truth value. Beside it, the formal language used in NARS has two more types of sentence: "question" and "goal". A *question* is either a statement whose truth value needs to be evaluated (a "yes/no" question), or a statement containing a variable to be instantiated (a "what" question). A *goal* is a statement whose truthfulness needs to be established by the system through the execution of some operations.

Each input sentence of the system is treated as a *task* to be processed:

**judgment.** An input judgment is a piece of new knowledge to be absorbed. To process such a task not only means to turn it into a belief of the system and add it into memory, but also means to use it and the existing beliefs to derive new beliefs.

**question.** An input question is a user query to be answered. To process such a task means to find a belief that answers the question as well as possible.

**goal.** An input goal is a user command to be followed, or a statement to be realized. To process such a task means to check if the statement is already true, and if not, to execute some operations to make the statement true.

Therefore, no matter which type a task belongs to, to process it means to interact it with the beliefs of the system, which is a collection of judgments, obtained or derived from the past experience of the system.

In each inference step, a task interact with a belief. If the task is a judgment, then a previously mentioned inference rule may be used, to derive a new judgment. This is called "forward inference". If the task is a question or a goal, "backward inference" happens as the following: A question $Q$ and a judgment $J$ will give rise to a new question $Q'$ if and only if an answer for $Q$ can be derived from an answer for $Q'$ and $J$, by applying a forward inference rule; a goal $G$ and a judgment $J$ will give rise to a new goal $G'$ if and only if the achieving of $G$ can be derived from the achieving of $G'$ and $J$, by applying a forward inference rule. Therefore, backward inference is the reverse of forward inference, with the same rules.

If a question has the form of "$? \circ\!\!\rightarrow P$", the system is asked to find a term that is a typical instance of $P$, according to existing beliefs. Ideally, the best answer would be provided by a belief "$S \circ\!\!\rightarrow P < 1, 1 >$". But this is impossible, because a confidence value can never reach 1 in NARS. Suppose the competing answers are "$S_1 \circ\!\!\rightarrow P < f_1, c_1 >$" and "$S_2 \circ\!\!\rightarrow P < f_2, c_2 >$", the system will choose the conclusion that has higher *expectation* value (Wang, 1994; Wang, 1995), defined as

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

Since the system usually has multiple goals at a given time, and they may conflict with each other, in NARS each goal has a "utility" value, indicating its "degree of desire". To relate utility values to truth values, a virtual statement $D$ is introduced for "desired state", and the utility of a goal $G$ is defined as the truth value of the statement "$G \Rightarrow D$", that is, the degree that the desired state is implied by the achieving of this goal. In this way, the functions needed to calculate utility values can be obtained from the truth-value functions.

In NARS, "decision making" happens when the system needs to decide whether to actually pursue a goal. If the goal directly corresponds to an operation of the system, then the decision is whether to execute it. This definition of decision making in NARS is different from that in traditional decision theory (Savage, 1954), in the following aspects:

- A goal is not a state of the world, but a statement in the system.
- The utility value of a statement may change over time when new information is taken into consideration.
- The likelihood of an operation to achieve a goal is not specified as a probability, but as a truth value defined above.
- The decision is on whether to pursue a goal, but not on which goal is the best one in a complete set of mutually-exclusive goals.

The decision depends on the expected utility value of the goal. Clearly, if more negative evidence than positive evidence is found when evaluating whether a goal is desired, the goal should not be pursued.

The above description shows that, instead of distinguishing "intention" and "desire" (Cohen and Levesque, 1990; Rao and Georgeff, 1995), in NARS the commitment to each goal is a matter of degree, partially indicated by the utility value of the goal. This kind of commitment is related to the system's beliefs, and is adjusted constantly according to the experience of the system, as part of the inference activity.

## Memory and control

Since in NARS no belief is absolutely true, the system will try to use as many beliefs as possible to process a task, so as to provide a better (more confident) solution. Due to insufficient resources, the system cannot use all relevant beliefs for each task. Since new tasks come from time to time, and the system generates derived tasks constantly, at any moment the system typically has a large amount of tasks to process.

For this situation, it is too rigid to set up a static standard for a satisfying solution (Strosnider and Paul, 1994), because no matter how careful the standard is determined, sometimes it will be too high, and sometimes too low, given the ever changing resources demand of the existing tasks. What NARS does is to try to find the best solution given the current knowledge and resources restriction, similar to what an "anytime algorithm" does (Dean and Boddy, 1988).

NARS *distributes* its processing power among the tasks in a time-sharing manner, meaning that the processor time is cut into fixed-size slices, and in each slice a single task is processed. Because NARS is a reasoning system, its processing of a task divides naturally into inference steps, one per time-slice.

In each inference step, a task is chosen probabilistically, and the probability for a task to be chosen is proportional to its priority value, a real number in [0, 1]. As a result, priority determines the processing speed of a task. At a given moment, if the priority of task $t_1$ is $u_1$ and the priority of task $t_2$ is $u_2$, then the amounts of time resources the two tasks will receive in the near future keep the ratio $u_1 : u_2$. Priority is therefore a relative rather than an absolute quantity. Knowing that $u_1 = 0.4$ tells us nothing about when task $t_1$ will be finished or how much time the system will spend on it. If $t_1$ is the only task in the system, it will get all of the processing time. If there is another task $t_2$ with $u_2 = 0.8$, the system will spend twice as much time on $t_2$ as on $t_1$.

If the priority values of all tasks remain constant, then a task that arises later will get less time than a task that arises earlier, even if the two have the same priority value. A natural solution to this problem is to introduce an "aging" factor for the priority of tasks, so that all priority values gradually *decay*. In NARS, a real number in (0, 1), called *durability*, is attached to each priority value. If at a given moment a task has priority value $u$ and durability factor $d$, then after a certain amount of time has passed, the priority of the task will be $ud$. Therefore, durability is a relative measurement, too. If at a certain moment $d_1 = 0.4$, $d_2 = 0.8$, and $u_1 = u_2 = 1$, we know that at this moment the two tasks will get the same amount of time resources, but when $u_1$ has decreased to 0.4, $u_2$ will only have decreased to 0.8, so the latter will then be receiving twice as much processing time as the former.

By assigning different priority and durability values to tasks, the user can put various types of time pressure on the system. For example, we can inform the system that some tasks need to be processed immediately but that they have little long-term importance (by giving them high priority and low durability), and that some other tasks are not urgent, but should be processed for a longer time (by giving them low priority and high durability).

To support priority-biased resource allocation, a data structure called "bag" is used in NARS. A bag can contain certain type of items with a constant capacity, and maintain a priority distribution among the items. There are three major operations defined on bag:

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

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

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

NARS runs by repeatedly executing the following working cycle:

1. Take a concept from the memory by priority.
2. Take a task from the task bag of the concept by priority.
3. Take a belief from the belief bag of the concept by priority.
4. According to the combination of the task and the belief, call the applicable inference rules on them to derive new tasks.
5. Adjust the priority of the involved task, belief, and concept, according to how they behave in this inference step, then put them back into the corresponding bags.
6. Put the new (input or derived) tasks into the corresponding bags, and create a belief for each task that is a judgment. If a new belief provides the best solution so far for a user-assigned task, report a solution to the user.

In the above step 5, the priority value of each item reflects the amount of resources the system plans to spend on it in the near future. It has two factors:

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

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

In NARS the processing of tasks are interwoven, even when they are not directly related to each other in contents. The starting and ending point of a task processing are not clearly defined, because the system never waits for new tasks in a special state, and it never reports a final answer, then stop working on a task right after it. What it does to a task is strongly influenced by the existence of other tasks.

NARS runs continuously, and has a "life-time of its own" (Elgot-Drapkin et al., 1991). When the system is experienced enough, there will be lots of tasks for the system to process. The system's behavior will to a certain extent depend on *its own* tasks, which are more or less independent of the original tasks assigned by the users, even though historically derived from them. This is the *functional autonomy* phenomena (Allport, 1937; Minsky, 1985).

NARS processes many tasks in parallel, but with different speeds. This "controlled concurrency" control mechanism is similar to Hofstadter's "parallel terraced scan" strategy (Hofstadter and FARG, 1995) and the resources allocation in genetic algorithms. In NARS, how a task is processed depends on the current beliefs, as well as the priority distribution among concepts, tasks, and beliefs. Since these factors change constantly, the solution the system finds for a task is context dependent.

## Discussion

The above description shows that the major components of NARS are fundamentally different from that of conventional reasoning systems. To discuss these differences in detail is beyond the scope of this paper. Some of such discussions can be found in the previous publications on NARS, and more will come in a book in preparation which covers the whole NARS project (Wang, 2004).

The only topic to be discussed here are the desired properties of an intelligent core in an integrated AI system, introduced previously in the paper.

NARS is designed according to a theory of intelligence that is consistent with many research results of psychology, philosophy, and linguistics. Few people doubt that the human mind is adaptive, and is able to work with insufficient knowledge and resources. Many hard problems in AI also need to be solved under this restriction. Actually, for a given problem, "having sufficient knowledge" means that we have an algorithm as its solution, and "having sufficient resources" means that the algorithm can be run in a computer system to solve each instance of the problem in realistic situations. If both conditions are satisfied, the problem can be solved as conventional computer programming. "Intelligence" is needed only when the above conditions cannot be satisfied, and traditional theories do not work (because they all assume the sufficiency of knowledge and/or resources in this or that way).

This is especially true for an integrated AI system. For a system equipped with various tools to be called "intelligent", it must be able to deal with novel problems in real time, here "novel problems" are exactly those for them the system's knowledge is insufficient, and "real time" means that the system cannot afford the time to explore all possibilities. In such a situation, being intelligent means to use available knowledge and resources as efficiently as possible by learning from the past experience of the system.

NARS is general and flexible enough for the unification of various cognitive facilities. In the system, reasoning, learning, and categorization are different aspects of the same underlying processes (Wang, 2000; Wang, 2002). With the addition of procedural interpretation of statements, problem solving, planning, and decision making are integrated into the system. Several prototypes of the system have been implemented, and so the basic ideas have been shown to be feasible.

Though NARS per se does not include any other AI techniques are components, it can be used as an intelligent core of an integrated system. A program outside the core will correspond to an operation that can be invoked by NARS, and the results of the program will be new tasks for the system. Knowledge about the outside programs will be represented as beliefs on the preconditions and consequences of the operations, as described previously. This kind of knowledge can be directly provided to the system by the users, and/or learned by the system from its own experiences. Such outside program can include domain-specific tools, as well as general-purpose modules for natural language interface, sensorimotor mechanism, and so on.

## References

Allen, J. F. (1984). Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154.

Allport, G. (1937). The functional autonomy of motives. *American Journal of Psychology*, 50:141–156.

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

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

Brooks, R. (1991). Intelligence without representation. *Artificial Intelligence*, 47:139–159.

Cohen, P. and Levesque, H. (1990). Intention is choice with commitment. *Artificial Intelligence*, 42:213–261.

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

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

Englebretsen, G. (1996). *Something to Reckon with: the Logic of Terms*. Ottawa University Press, Ottawa.

Feigenbaum, E. and McCorduck, P. (1983). *The Fifth Generation : Artificial Intelligence and Japan's Computer Challenge to the world*. Addison-Wesley Publishing Company, Reading, Massachusetts.

Frege, G. (1970). Begriffsschrift, a formula language, modeled upon that of arithmetic, for pure thought. In van Heijenoort, J., editor, *Frege and Gödel: Two Fundamental Texts in Mathematical Logic*, pages 1–82. Harvard University Press, Cambridge, Massachusetts.

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

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

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

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

Piaget, J. (1963). *The Origins of Intelligence in Children*. W.W. Norton & Company, Inc., New York. Translated by M. Cook.

Rao, A. S. and Georgeff, M. P. (1995). BDI-agents: from theory to practice. In *Proceedings of the First Intl. Conference on Multiagent Systems*, San Francisco.

Savage, L. (1954). *The Foundations of Statistics*. Wiley, New York.

Stork, D. (1997). Scientist on the set: An interview with Marvin Minsky. In Stork, D., editor, *HAL's Legacy: 2001's Computer as Dream and Reality*, pages 15–30. MIT Press, Cambridge, Massachusetts.

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

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

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

Wang, P. (2000). The logic of learning. In *Working Notes of the AAAI workshop on New Research Problems for Machine Learning*, pages 37–40, Austin, Texas.

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

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

Wang, P. (2002). The logic of categorization. In *Proceedings of the 15th International FLAIRS Conference*, Pensacola, Florida.

Wang, P. (2004). Rigid Flexibility — The Logic of Intelligence. Manuscript in preparation.

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