From NARS to a Thinking Machine

Pei WANG
Temple University, Philadelphia, USA
http://www.cis.temple.edu/~pwang/

Abstract. NARS is an AGI project developed in the framework of reasoning system, and it adapts to its environment with insufficient knowledge and resources. The development of NARS takes an incremental approach, by extending the formal model stage by stage. The system, when finished, can be further augmented in several directions.

Keywords. General-purpose system, reasoning, intelligence, development plan

1. Introduction

NARS (Non-Axiomatic Reasoning System) is a project aimed at the building of a general-purpose intelligent system, or a "thinking machine". This chapter focuses on the methodology of this project, and the overall development plan to achieve this extremely difficult goal. There are already many publications discussing various aspects of NARS, which are linked from the author's website. Especially, [1] is a recent comprehensive description of the project.

If there is anything that the field of Artificial Intelligence (AI) has learned in its fifty-year history, it is the complexity and difficulty of making computer systems to show the "intelligence" as displayed by the human mind. When facing such a complicated task, where should we start? Since the only known example that shows intelligence is the human mind, it is where everyone looks at for inspirations at the beginning of research. However, different people get quite different inspirations, and consequently, take different approaches to AI.

The basic idea behind NARS can be traced to the simple observation that "intelligence" is a capability possessed by human beings, but not by animals and traditional computer systems. Even if we accept the opinion that some animals and computers also show intelligence, they are still far inferior in this ability when compared to human beings.

If we can indeed draw the boundary of "intelligent system" to include normal humans, but not typical animals and computer systems, then the next question is: what is the difference between these two types of system? Hopefully the answer to this question can tell us what intelligence really is.

Even if this boundary of the category of intelligence is accepted, there are still many differences that can be found and justified. Now the problem becomes to identify a small number of them that are "essential", in the sense that they can derive or explain many other differences.

The design of NARS is based on the belief that the essence of intelligence is *the capability to adapt to the environment and to work with insufficient knowledge and resources* [2]. Therefore, an intelligent system should rely on constant processing capacity, work in real time, open to unexpected tasks, and learn from experience.

According to this opinion, whether a system is intelligent, or how intelligent it is, is not determined by what the system *can do*, but by what it *can learn to do*. An intelligent system does not need to be identical to the human brain in internal structure, or the human mind in external behaviors, nor does it have to provide optimal solutions to certain practical problems. Instead, the system should be general, flexible, and creative.

Though the above understanding of intelligence sounds simple, it does explain many phenomena observed in the human mind [1].

Similar to its conceptual basis, the engineering plan of NARS also follows "minimalism", in the sense that the goal is not to maximize the system's performance, but to minimize its theoretical assumption and technical foundation, while still being able to realize the conception of intelligence introduced above. The scientific reason for following this approach is that it produces a precise and verifiable theory; the practical reason is that it leads to a manageable project under the restriction of available development resources.

NARS is built in the form of a reasoning system, with a *language* for knowledge representation, a *semantic theory* of the language, a set of inference *rules*, a *memory* structure, and a *control* mechanism. The first three components are usually referred to as a *logic*. The reasoning framework has the advantage of being domain-independent, and combining the justifiability of individual inference steps and the flexibility of linking these steps together in a context-sensitive manner in run time.

In the following, the development plan of NARS is introduced first, which shows how the system is built stage by stage. After that, several important but optional augmentations of NARS are introduced and analyzed. Finally, several topics about the development of this system are discussed.

2. NARS Roadmap

The development of NARS takes an incremental approach consisting four major stages. At each stage, the logic is extended to give the system a more expressive language, a richer semantics, and a larger set of inference rules; the memory and control mechanism are then adjusted accordingly to support the new logic. Consequently, the system becomes more intelligent than at the previous stage, according to the previous conception of intelligence, by being more adaptive and more efficient in using available knowledge and resources.

The following description omits technical details and comparisons with other systems, which can be found in the other publications on NARS, such as [1].

2.1 Basic reasoning

At this stage, the system is equipped with a minimum *Non-Axiomatic Logic*.

The logic uses a *categorical* language called "Narsese", in which every statement consists of a *subject term* and a *predicate term*, linked by an *inheritance relation*. Intuitively, the statement says that the subject is a *specialization* of the predicate, and the predicate is a *generalization* of the subject. For example, "*bird* → *animal*" is such a statement, with "*bird*" as the subject term and "*animal*" as the predicate term. A "term", at the current stage, is just an atomic identifier. The inheritance relation is defined as from one term to another term, and as being reflexive and transitive in idealized situations.

For a given term, its *extension* is the set of its known specializations, its *intension* is the set of its known generalizations, and its *meaning* consists of its extension and intension. Therefore, given inheritance statement "*bird* → *animal*", "*bird*" is in the extension of "*animal*", and "*animal*" is in the intension of "*bird*".

It can be proved that a statement is true if and only if the extension of its subject is a subset of the extension of its predicate, and, equivalently, if the intension of its predicate is a subset of the intension of its subject.

Based on this result, statements can be made multi-valued to represent incomplete inheritance. For a given inheritance statement, its *positive evidence* is defined as the set of the terms that are either in both the extension of the subject and the extension of the predicate, or in both the intension of the predicate and the intension of the subject; its *negative evidence* is defined as the set of the terms that are either in the extension of the subject but not the extension of the predicate, or in the intension of the predicate but not the intension of the subject. Evidence is defined in this way, because as far as a piece of positive evidence is concerned, the statement is true; as far as a piece of negative evidence is concerned, the statement is false.

For a given statement, if the amounts of positive and total (i.e., positive plus negative) evidence are written as w^+ and w, respectively, then the truth-value of the statement is represented as a pair of real numbers in [0, 1], <*frequency*, *confidence*>, where $frequency = w^+ / w$, and $confidence = w / (w + 1)$. Consequently, truth-value in NARS uniformly represents several types of uncertainty, including *randomness*, *fuzziness*, and *ignorance* [1].

Defined in this way, Non-Axiomatic Logic uses an "experience-grounded" semantics, where the meaning of each term and the truth-value of each statement are determined according to the relevant experience of the system, by the system itself.

A statement with truth-value is called a "judgment". The inference rules of NARS are defined on judgments, and are designed and justified according to the above experience-grounded semantics. A typical inference rule is *syllogistic*, that is, it takes a pair of inheritance judgments as premises, which share exactly one common term, and the conclusion is an inheritance judgment between the other two terms. The position of the shared term determines the type of the inference, including the following (truth-values omitted):

Table 1.

	Deduction	**Abduction**	**Induction**
Premise 1	M → P	P → M	M → P
Premise 2	S → M	S → M	M → S
Conclusion	S → P	S → P	S → P

Each inference rule has an associated truth-value function, which calculates the truth-value of the conclusion from those of the premises, by determining the amount evidential support the conclusion gets directly from the evidence provided by the premises. An inference rule is *valid* if it can be justified according to the experience-grounded semantics. The details and justifications of the rules can be found in [1] and other publications on NARS.

There is a special rule used for *revision*, where the two premises contain the same statement, but are supported by evidence collected from different sources. The revision rule produces a conclusion, with the same statement as content, and a truth-value corresponding to the accumulated evidence from both sources.

A *belief* of the system is a judgment coming from, or derived according to, the system's experience. At this stage, a *task* that the system can carry out is either a question to be answered, or a piece of new knowledge (as a judgment) to be absorbed.

A belief can be used to answer questions, both of the "yes/no" type and the "what" type, as well as to derive new beliefs following the inference rules. For a given question, the syllogistic rules can also be used *backward* to produce derived questions, whose answers will lead to the answer of the original question, using *forward* inference.

Roughly speaking, the system's memory consists of a belief network, in which each node is named by a term, and each link corresponds either to a belief, or to a task. Each node with its incoming and outgoing links forms a *concept*.

When the system is running, usually there are many tasks under processing in parallel. The system assigns a *priority-value* to every concept, task, and belief. At each inference step, a concept is selected, and then a task and a belief are selected within the concept. All these selections are *probabilistic*, that is, an item with a higher priority-value has a higher probability to be selected.

With the selected task and belief as premises, applicable inference rules derive new tasks (and new beliefs, if the task is a judgment), and add them into the memory. When the memory is full, items (beliefs, tasks, and concepts) with the lowest priority values are removed to release their previous occupied space.

After each step, the priority-values of the involved concept, task, and belief are adjusted, according to the immediate feedback obtained in this step. In the long run, higher priority-values will be obtained by items that are more useful in the past history and more relevant in the current context.

For a given task, the system's processing course is a sequence of inference steps. Such a sequence is formed at the run time, does not depend on any predetermined task-specific algorithm, but is determined by many factors in the past experience and current context of the system. Since these factors change constantly, the system's processing path and response to a task is usually not a function of the task, and they are neither fully predictable in advance nor fully repeatable afterwards, given the task alone.

Designed in this way, the system is indeed adaptive and can work with insufficient knowledge and resources, though it can only handle simple tasks, because of its limited expressive and inferential power at this stage.

2.2 First-order reasoning

At the second stage, Non-Axiomatic Logic is extended by adding compound terms and variants of the inheritance relation into the language and the rules.

The *similarity relation*, "↔", is a variant of the inheritance relation ("→"), and can be seen as symmetric inheritance. For a statement containing two terms linked by the similarity relation, the *positive evidence* includes the terms in the extension (or intension) of both terms, and the *negative evidence* includes the terms in the extension (or intension) of one term but not the other.

Inference rules involving similarity include *comparison* (such as to derive "S ↔ P" from "S → M" and "P → M") and *analogy* (such as to derive "S → P" from "S ↔ M" and "M → P"). Again, each rule has an associated truth-value function.

A *compound term* is formed by an *operator* from one or more *component terms*, and the composing process can be repeated to build compound terms with complicated internal structures.

The meaning of a compound term has a literal part and an empirical part. The former comes from the meaning of its components in a way determined by the operator; the latter comes from the role the compound term as a whole plays in the experience of the system, just like how the meaning of an atomic term is determined at the previous stage. In different compound terms, these two parts have different relative significances. The more the system knows about a compound term, the less it is used in its literal meaning.

The following compound terms are introduced into Narsese at this stage:

Sets. Though a term is not defined as a set in general, there are special compound terms corresponding to sets:

- An extensional set is defined by exhaustively listing its instances, such as in "$\{T_1, T_2, \ldots, T_n\}$".
- An intensional set is defined by exhaustively listing its properties, such as in "$[T_1, T_2, \ldots, T_n]$".

In both cases, "$T_1, T_2, \ldots, T_n$" are terms serving as the components of the compound, and "{}" and "[]" are operators for extensional set and intensional set, respectively.

Intersections and Differences. Compound terms can be formed via set operations on the extensions or intensions of existing terms. Using two different terms T_1 and T_2 as components, four compound terms can be defined in this way:

- "$(T_1 \cap T_2)$" is their extensional intersection. Its extension is the intersection of the extensions of T_1 and T_2, and its intension is the union the intensions of T_1 and T_2.
- "$(T_1 \cup T_2)$" is their intensional intersection. Its intension is the intersection of the intensions of T_1 and T_2, and its extension is the union the extensions of T_1 and T_2.
- "$(T_1 - T_2)$" is their extensional difference. Its extension is the difference of the extensions of T_1 and T_2, and its intension is the intension of T_1.
- "$(T_1 \varnothing T_2)$" is their intensional difference. Its intension is the difference of the intensions of T_1 and T_2, and its extension is the extension of T_1.

Products and Images. To represent ordinary relations among terms that cannot be treated as *inheritance* or *similarity*, compound terms can be used so that the statement is still about inheritance. For example, an arbitrary relation R among terms A, B, and C can be represented as an inheritance statement "$(\times\ A\ B\ C) \rightarrow R$", where the subject term is a *product* "$(\times\ A\ B\ C)$", which is a compound with three components in the given order, and the symbol "$\times$" is the *product operator*. To isolate each component out of the compound term, the above inheritance statement can be equivalently rewritten into any of the following form:

- "$A \rightarrow (\perp\ R\ \Diamond\ B\ C)$"
- "$B \rightarrow (\perp\ R\ A\ \Diamond\ C)$"
- "$C \rightarrow (\perp\ R\ A\ B\ \Diamond)$"

In each of the three inheritance statements, the predicate term is an *extensional image*, with "$\perp$" as the operator, and "$\Diamond$" as a placeholder indicating the location of the subject in the relation. Symmetrically, there is also a type of compound term called *intensional image*.

For each type of compound term defined above, there are corresponding inference rules for the *composition* and *decomposition* of compound terms. Consequently, the inference process not only produces new tasks and beliefs, but also generates new terms and concepts.

At this stage, the memory architecture and control mechanism of the system are revised to uniformly handle *compound* terms and *atomic* terms.

2.3 Higher-order reasoning

Higher-order reasoning allows statements to be used as terms. At this stage, the system can carry out reasoning on higher-order statements, i.e., statements of statements.

In the simplest situation, a higher-order statement is formed by an ordinary relation that takes a statement as argument. Such relations include "believe", "know", "say", etc. For example, in Narsese "Birds are animals" can be represented as a (first-order) statement "*bird* → *animal*", and "Peter knows that birds are animals" as a higher-order statement "(× {*Peter*} {*bird* → *animal*}) → *know*".

Another way to form higher-order statements is to use statement operators *conjunction* ("∧"), *disjunction* ("∨"), and *negation* ("¬"), as in prepositional logic, except that in Narsese the statements are multi-valued, rather than binary. Each of the three operators has an associated truth-value function when used to form compound statements.

Two new relations are introduced between two statements: *implication* ("⇒", intuitively meaning "if") and *equivalence* ("⇔", intuitively meaning "if-and-only-if"). In the implication statement "S ⇒ P", S is a *sufficient condition* of P, and P is a *necessary condition* of S. In the equivalence statement "S ⇔ P", S and P are *sufficient and necessary conditions* of each other.

Implication ("⇒") and *equivalence* ("⇔") are defined to be isomorphic to *inheritance* ("→") and *similarity* ("↔"), respectively. Consequently, some of the rules of higher-order inference are isomorphic to certain rules of first-order inference:

Table 2.

	Deduction	**Abduction**	**Induction**
Premise 1	M ⇒ P	P ⇒ M	M ⇒ P
Premise 2	S ⇒ M	S ⇒ M	M ⇒ S
Conclusion	S ⇒ P	S ⇒ P	S ⇒ P

Each of these rules uses the same truth-value function as its first-order counterpart.

There are also inference rules that are specially designed for higher-order inference, and have no direct counterpart in first-order inference.

The terms in Narsese defined so far are *constant*, in the sense that each term uniquely indicates a concept in the system. To extend the expressing power of the language, *variable terms* are introduced at this stage.

There are two types of variable: an *independent variable* is named by an identifier with a prefix "#", and indicates a unspecific term in the extension or intension of another term; a *dependent variable* is named by an identifier with a prefix "#" and followed by a list (which may be empty) of independent variables, and indicates a specific-but-unnamed term in the extension or intension of another term.

The scope of a variable term is one statement, so that the same variable term name in different statements may indicate different (constant) terms in the system.

A statement may contain multiple variables with embedded scopes. For example, the following sentences can be represented in Narsese using two variables, with difference relationships:

- "*Every key can open every lock"* is represented as:
  "(({#x} → key) ∧ ({#y} → lock)) ⇒ ((× {#x} {#y}) → open)"
- "*Every key can open a lock*" is represented as:
  "({#x} → key) ⇒ (({#y(#x)} → lock) ∧ ((× {#x} {#y(#x)}) → open))"
- "*There is a key that can open every lock*" is represented as:
  "({#x()} → key) ∧ (({#y} → lock) ⇒ ((× {#x()} {#y}) → open))"
- "*There is a key that can open a lock*" is represented as:
  "({#x()} → key) ∧ ({#y()} → lock) ∧ ((× {#x()} {#y()}) → open)"

There are inference rules responsible for the substitution between variable terms and constant terms in inference steps.

Again, the memory architecture and control mechanism are revised to uniformly handle *first-order* inference and *higher-order* inference.

At this stage, the expressive power of Narsese is comparable to the language of predicate calculus, in the sense that the two roughly overlap to a large extent, though neither is a subset of the other. The inferential power of the logic is extended to cover conditional inference, hypothetical inference, and abstract inference. Different from predicate calculus, all inference rules in NARS are designed according to the assumption of insufficient knowledge and resources, and justified according to the experience-grounded semantics described previously.

2.4 Procedural reasoning

Procedural reasoning means to infer about events, operations, and goals.

The first step is to introduce *time* into the logic. In the previous stages, the truth-value of a statement is timeless. At this stage, an *event* is defined as a special type of statement that has temporal truth-value, that is, its truth-value can change over time.

In Narsese, the temporal attribute of an event is represented *relatively*, with respect to another event. For two events, the simplest temporal relations are either they happen at the same time, or one of them happens before the other. The other temporal relations between them can be represented by dividing the involved events into sub-events, and further specifying the temporal relations among these sub-events.

By combining the two primary temporal relations with existing operators and relations, new operators and relations are formed, such as *sequential conjunction* (","), *parallel conjunction* (";"), *predictive implication* ("/⇒"), *retrospective implication* ("\⇒"), and *concurrent implication* ("|⇒"). For example, "S /⇒ P" indicates that S is a *sufficient precondition* of P, and P is a *necessary postcondition* of S.

With this representation, temporal inference can be carried out by separately processing the logical relation and the temporal relation in the premises, then combining the two results in the conclusion, such as in the following rules:

Table 3.

	Deduction	**Abduction**	**Induction**

Premise 1	M /⇒ P	P \⇒ M	M /⇒ P
Premise 2	S /⇒ M	S /⇒ M	M \⇒ S
Conclusion	S /⇒ P	S /⇒ P	S /⇒ P

O*peration* is a special type of event, which the system can actively *execute*, not just passively *observe*. In other words, an operation is a statement under *procedural interpretation*, as in logic programming. With the operators and relations introduced previously, the system can have beliefs on the preconditions (including causes) and postconditions (including consequences) of a given operation, and carry out explaining, predicting, planning, and skill learning, by reasoning on these beliefs.

The execution of an operation is usually accomplished by a mechanism outside the reasoning system, such as a hardware device with certain sensor/effecter capability, or another computer program with certain information-processing capability. The reasoning system interacts with them by issuing execution commands and collecting execution consequences, and these activities consist of the sensorimotor processes of the system as a whole. Operations toward the outside of the system correspond to *perception and control* of the environment; operations toward the inside of the system correspond to *self-perception and self-control*.

Operations are the means for the system to achieve *goals*. Before this stage, NARS can accomplish two types of tasks: knowledge to be absorbed, and questions to be answered. At this stage, a goal is defined as a statement to be made true (or as close to true as possible) by executing proper operations. A goal can be seen as an event, whose preconditions and postconditions will be gradually revealed through reasoning on available knowledge, and eventually related to operations.

The system uses a *decision-making* procedure to create new goals from desirable and achievable events. For this purpose, it attaches a *desirability-value* to each event, and includes their calculation as part of the inference process.

Furthermore, the memory architecture and control mechanism are revised to uniformly handle *goals/operations* and ordinary *tasks/beliefs*.

3. NARS and Beyond

The development of NARS has been carried out for more than two decades, roughly according to the plan described above. At the current time, the first three stages have been mostly finished, and the last stage should be completed in a few years. On-line demonstrations with working examples are available at the author's website.

After throughout testing and tuning as described in [1], NARS will be a general-purpose reasoning system, built according to the conception of intelligence as the ability to adapt with insufficient knowledge and resources.

However, that will not be the end of this research adventure. According to the above conception of intelligence, it is always possible to make a system more intelligent by extending its input/output channels to enrich its experience, or by improving its efficiency when using available knowledge and resources.

After NARS *per se* is fully built, there are still several major ways to augment the capability of the system. In the following, each of them will be briefly described, as well as analyzed to explain why it is not considered as a necessary part of NARS.

3.1 Sensors and effectors

For NARS, "sensors and effectors" are the input/output devises that allow the system to directly interact with the environment outside the language channel.

Such a device can be plugged into the system, according to the *operation* mechanism introduced in the previous section. Concretely, each usage of the device can be triggered by a command in the form of "(op, a_1, …, a_n)", where "op" is an operator (the action to be performed), and "a_1, …, a_n" are arguments of the operation (information needed in performing the action). To the reasoning system, this operation is inheritance statement "$(\times \{a_1\} \ldots \{a_n\}) \rightarrow op$" under procedural interpretation.

As explained before, the beliefs about an operation are represented mostly as its (pre and post) conditions. When the system concludes that a goal can be achieved by this operation, its degree of desirability is increased. If the operation is judged by the decision-making mechanism as sufficiently desirable and feasible, it becomes a goal itself, and it will be achieved directly by issuing the corresponding command, so that the device will perform the action to the (internal or external) environment. The reasoning system then will collect the feedback using its sensors to check if the goal has indeed been achieved, and to decide what to do next. A sensor is also invoked by corresponding operations, and it produces Narsese judgments, each of which states that an experienced item or pattern can be generalized by another one to a certain extent.

Because of the insufficiency of knowledge and resources, the system usually never knows all the preconditions and postconditions of each operation, and its expectations are not always confirmed by its future experience. Nor can it be guaranteed that the executed operations correspond to the optimal way to achieve its goals. Instead, the system just does its best, under the restriction of available knowledge and resources.

NARS is designed as a general-purpose system, without innate problem-specific tools, but these tools can be plugged into the system, and will be handled in a uniform manner. Consequently, NARS can be used either as an "intelligent operating system" with various software tools, or a "mind" of a robot with various hardware devices.

In artificial intelligence and cognitive sciences research, some authors stress the importance of sensorimotor to intelligence and cognition, and argue that the mind is situated and embodied [3, 4]. These opinions are agreeable, as far as they are taken to mean that thinking must be grounded in *experience*. However, it does not mean that thinking must be grounded in *human sensorimotor experience*. Since NARS is designed according to an experience-grounded semantics, it is situated and embodied. However, because the system is not designed to duplicate concrete human behaviors and capabilities, it is not equipped with human sensors and effecters. For example, there is no doubt that vision plays a central role in human cognition, and that computer vision has great practical value, but it does not mean that vision is needed in every intelligent system. Because of these considerations, sensors and effecters are treated as optional parts of NARS.

3.2 Natural languages

As mentioned previously, NARS uses Narsese, a formally defined language, to communicate with its environment. Since the language uses an experience-grounded semantics, the truth-value of each statement and the meaning of each term in the language are determined by the system's relevant experience. In this aspect, the language is very similar to natural languages.

On the other hand, since the grammar of Narsese is formally defined, it is very different from the grammar of any natural language. For NARS to use a natural language to communicate with human users, it needs to learn the grammar and lexicon of the language.

In NARS, natural-language learning can be carried out by the system's general-purpose learning mechanism on linguistic materials. Each word, phrase, and sentence of a natural language will be represented within the system by a term. These terms have a many-to-many mapping to the terms directly used in the system's "native language", Narsese, and this mapping corresponds to a *symbolize* relation in Narsese. The truth-value of a symbolizing statement indicates the frequency and confidence for the word/phrase/sentence (in the natural language) to be used as the *symbol* of the term (in Narsese), according to the experience of the system.

In language understanding process, NARS will not have separate parsing and semantic mapping phases, like in many other natural language processing systems. Instead, for an input sentence, the recognition of its syntactic structure and the recognition of its semantic structure will be carried out hand-in-hand. The process will start by checking whether the sentence can be understood as a whole, as the case of proverbs and idioms. If unsuccessful, the sentence will be divided recursively into phrases and words, whose sequential relations will be tentatively mapped into the structures of compound terms, with components corresponding to the individual phrases and words. If there are multiple candidates in the mapping process (i.e., ambiguity), the truth-values of the resulting statements will show which one is better supported, according to the system's experience.

The language production process is roughly the reverse of the understanding process. Driven by the goal of the communication activity, the ideas to be expressed will be selected or generated, then "translated" from Narsese into the target natural language, according to the learned symbolizing relation between the two languages.

In this way, NARS has the potential to learn and use any natural language, as far as its experience *in* that language can be related to its experience *outside* that language. However, due to the inevitable difference in experience, the system cannot always be able to use a natural language as a native speaker. Even so, its proficiency in that language should be sufficient for many practical purposes.

Being able to use any natural language is not a necessary condition for being intelligent. Since the aim of NARS is not to accurately duplicate human behaviors so as to pass the Turing Test [5], natural language processing is optional for the system.

3.3 Education

NARS processes tasks using available knowledge, though the system is not designed with a ready-made knowledge base as a necessary part. Instead, all the knowledge, in principle, should come from the system's experience. In other words, NARS as designed is like a baby that has great potential, but little instinct.

For the system to serve any practical purpose, extensive *education*, or *training*, is needed, which means to build a proper internal knowledge base (or call it belief network, long-term memory, etc.) by feeding the system with certain (initial) experience.

Various knowledge sources will be made available for NARS, and the possibilities include (though not limited to):

- Existing knowledge bases. NARS can be connected to existing knowledge bases, such as Cyc (for commonsense knowledge), WordNet (for linguistic

knowledge), Mizar (for mathematical knowledge), and so on. For each of them, a special interface module should be able to approximately translate knowledge from its original format into Narsese.

- The Internet. It is possible for NARS to be equipped with additional modules, which use techniques like semantic web, information retrieval, and data mining, to directly acquire certain knowledge from the Internet, and put them into Narsese.
- Natural language interface. After NARS has learned a natural language (as discussed previously), it should be able to accept knowledge from various sources in that language.

Additionally, interactive tutoring will be necessary, which allows a human trainer to monitor the establishing of the knowledge base, to answer questions, to guide the system to form a proper goal structure and priority distributions among its concepts, tasks, and beliefs.

Unlike most of the training processes currently studied in machine learning, NARS cannot be trained to converge to certain predetermined stimulus-response mappings. Instead, the process will be more like the education of human beings, where learning is open-ended, and the tutors will have strong influence, but not complete control, of the system's behaviors. Therefore, the education theory for NARS will be similar to, though not necessarily identical to, that for human beings.

To improve the efficiency of education, it is possible to copy the memory of a trained NARS system into other NARS systems, so that they will not need to repeat the training process. Also, it is possible to directly edit the memory of a system to modify or implant certain knowledge. In theory, all of these shortcuts should be equivalent to certain possible experience of the system, so they do not conflict with the principle that all the knowledge of NARS comes, directly or indirectly, from experience.

One important issue to be handled through education is ethics. Unlike argued by some other researchers, NARS is not an attempt to design a "friendly AI". As far as its initial state is concerned, the system is ethically neutral, since it can has any beliefs and goals. To make a NARS implementation "human friendly" means to give it certain beliefs and goals, which is an education mission, not a design mission. Even if something like Asimov's "Three Laws of Robotics" is implanted into the system's memory (which is possible), it still cannot fully control the system's behaviors, due to the insufficiency of knowledge and resources in the system. What should and can be done is to educate an AI system like a child, by controlling its initial experience, to make it to love human beings, and to understand how to benefit the human beings, as far as its intelligence can tell.

NARS is designed according to the belief that the essence of "intelligence" is in the ability of flexibly acquiring, deriving, and applying knowledge, but not in possessing a large amount of human knowledge (as argued in [6]). For this reason, the building of a knowledge base for a specific practical application is an optional augmentation of the system.

3.4 Socialization

If multiple copies of NARS are implemented with different system parameters, hardware/software, and experience, they will form different beliefs and goals, and behave differently. However, as soon as they begin to communicate with each other, they will have common experience, and some consensus will be developed among the

systems. Furthermore, these systems may begin to cooperate in the solving of certain complicated problems, as well as to compete for certain resources.

Since NARS uses the same language, Narsese, for input and output, the current design needs no major change to allow communication in a multi-system environment, which will lead to *socialization*, a continuing process whereby the system form and adjust its beliefs and goals as a consequence of interaction with other systems.

Like for human mind, socialization will play an important role in the mental development of an AI system, which is different from the role played by education, though both happen as consequences of communication. In education, the involved systems (the trainee and the trainer) have asymmetric status, while in socialization all the involved systems have similar status.

For an implemented NARS system, many concepts and skills can only be fully acquired through socialization, such as self/other distinction, speech action, game playing, and moral discipline.

Furthermore, with such a multi-system community as a model, many interesting topics can be studied. For example, since each system in the community is based on the same experience-grounded semantics, the dynamics of the community as a whole will show the forming and changing of common knowledge, which will provide insights about the evolution of language, science, and culture.

Even though socialization plays an important role in the growth of a mind (either natural or artificial), it is considered as optional for NARS. For a theoretical reason, it is because the notion of intelligence accepted in this project does not presume the existence of a society as part of the environment; for a practical reason, it is because to consider such a society is not required when the system is designed. On the other hand, unless each single system is properly designed, it will not be very fruitful to study how a multi-system community behaves.

3.5 Hardware

As a reasoning system, the running process of NARS consists of individual inference steps. Since the system is built around a *term logic*, it has the property that each inference step happens within a "concept", which collects beliefs and tasks about the same term. As a result, a concept can be naturally taken as a processing unit, which manages its own knowledge and resources. Different concepts independently execute a similar routine to carry out an inference step, and pass messages (tasks) to one another to cooperate in the processing of the tasks in the system.

Such a parallel-and-localized processing mode allows NARS to achieve higher performance and efficiency by running on specially designed hardware. Roughly speaking, such a hardware system will consist of a large number of processing units, each with its own processor and memory, corresponding to a concept in NARS. Each processing unit only needs to implement a simple algorithm, as required by an inference step, plus the algorithms needed for inter-concept communication and resources management. Such an implementation will surely achieve much higher time efficiency, compared to the current implementation in a general-purpose computer with a single CPU.

It is important to realize that even in such a special hardware implementation, the basic principle of NARS remains the same: the system still needs to work with insufficient knowledge and resources. Since the number of processing unit is a constant, and so does the capacity of each unit, they will need to be shared by the concepts, because the system as a whole will producing new concepts from time to

time, whose number will soon exceed the number of processing units. Consequently, the system still need time-sharing and space-sharing, and it is only that what to be shared is not a single CPU and RAM, but many processing units.

Some people blame the von Neumann architecture of computer for the past failure of AI, but the argument is not convincing. It is true that the current computer architecture is not designed especially for AI, but it has not been proved that it cannot be used to implement a truly intelligent system. Special hardware is optional for NARS, since the system can be fully implemented on the current hardware/software platform, though special hardware will surely make it work better.

3.6 Evolution

Under the assumption of insufficient knowledge, all *object-level* knowledge in NARS can be modified by the system's various learning mechanisms. For example, the truth-value of any belief is revisable, and the priority-value of any belief, task, or concept is adjustable. All these changes are driven by the system's experience.

However, the current design of NARS does not support any automatic change of the *meta-level* knowledge of the system, which includes the grammar of Narsese, the factors determining truth-value and meaning, the inference rules, the resource-allocation mechanism, the priority-distribution strategy, as well as the values of system parameters. Meta-level knowledge is determined when the system is designed, and remains fixed during the system's life cycle, because any change at this level may destroy the system's consistency and integrity.

Obviously, a change in meta-level knowledge, that is, the system design, has the possibility of increasing the system's intelligence, though it is usually a dangerous experiment. For NARS, such changes are left for a separate *evolution process*, which will not be carried out by the reasoning/learning mechanisms in the current design.

The evolution of NARS will follow ideas similar to genetic algorithm [7]. First, a multi-dimensional design space will be determined, in which each dimension corresponds to a specific aspect of the system's design, and valid values on the dimension correspond to possible design choices for that aspect. In this way, a concrete system design can be represented by a point in the design space, whose coordinates form the "genetic code" of the system.

To search for the best design using genetic algorithm, a population of systems will be maintained, and in each generation, fitness values of the systems will be evaluated. When the next generation of system is produced, systems with higher fitness values will have a higher chance to be parents of new systems, which partially inherit the genetic codes of their parents. Some random mutation may also happen to the genetic code of a new system. In the long run, a natural selection process will produce systems with better fitness, which in this context means higher intelligence.

Though such an evolution process may indeed produce more intelligent NARS systems, it should not be confused with the experience-driven learning processes in NARS, which is what intelligence is about. In general, we should see *intelligence* and *evolution* as two different forms of *adaptation*. Roughly speaking, intelligence is achieved through experience-driven changes ("learning" or "conditioning") within a single system, while evolution is achieved through experience-independent changes ("crossover" or "mutation") across generations of systems. The "intelligent" changes are more justifiable, gradual, and reliable, while the "evolutionary" changes are more incidental, radical, and risky. Though the two processes do have some common properties, their basic principles and procedures are quite different.

4. Discussions

In this section, several issues about the development plan of NARS are discussed, in the context of the Artificial General Intelligence Research Institute Workshop.

4.1 Why to aim at a principle

The first major feature that distinguishes NARS from other AI projects is its conception of *intelligence* as the *principle* of *adaptation with insufficient knowledge and resources*. To most people, "intelligence" is more often associated with concrete behaviors, capabilities, and functions, than with general principles. This topic was initially addressed in [2], and has also been discussed in other publications on NARS, such as [1]. In the following, it is approached from a slightly different perspective.

As said at the beginning of the chapter, the boundary of "intelligent system" should be drawn in such a way, so as to include normal humans, but not typical animals and computer systems. On the other hand, we do not want to make the notion too narrow to take the human mind as the only possible form of intelligence. For example, to ask an AI system to be indistinguishable from human in behavior is too strong a request. Turing proposed his "Imitation Game" as a *sufficient* condition for a system to be intelligent, though he admitted that it might not be a *necessary* condition. He just thought that "if, nevertheless, a machine can be constructed to play the imitation game satisfactorily, we need not be troubled by this objection" [5]. Similarly, even though the human brain is indeed the only *known* way to produces intelligence, to take it as the only *possible* way to do that shows a lack of imagination.

On the other hand, to take intelligence as the ability of solving certain concrete problem seems to make it too easy, no matter how complicated the problem is. For example, the IBM computer system Deep Blue defeated the world chess champion, but many people still do not think it as intelligent, because it can do little beside chess. Indeed, almost all computer systems and animals can do something better than human beings, but they do not be called "intelligent" because of that. Otherwise, the concept of intelligence would be trivialized.

Many people enter the field of AI with a vague feeling that what we call "intelligence" in our everyday language has certain fundamental difference from how a conventional computer system works, and being more intelligent does not means to evaluate arithmetic expressions faster, to remember phone numbers for a longer period, or to accurately repeat a complicated procedures for more times. Instead, it is associated with features like generality, flexibility, and creativity. These features are displayed by an intelligent system in its problem-solving behaviors in various domains, but cannot, and should not, be bounded to a certain type of problem. Instead, they should be described as produced by a general *principle*, through a mechanism realizing the principle, which can be applied to various problems in various domains.

Such a "principle-based" conception of intelligence also helps for establishing the identity of a new research field, whether we choose to call it AI or AGI (Artificial General Intelligence). According to this definition, AI is different from theoretical Computer Science in that the latter mainly studies computer systems working with sufficient (though may be limited) knowledge and resources, while AI assume the opposite; AI is different from cognitive modeling in that the latter attempts to duplicate the details of human behavior, while AI only attempts to duplicate the general principle abstracted from concrete human behavior.

For practical reasons, almost all problems currently labeled as "AI" share the common nature that they need to be solved by adaptive systems working with insufficient knowledge and resources, and what we do not have yet is a normative theory for how to build such systems.

4.2 Why to take a unified approach

Intelligence, as displayed in the human cognition process, can be studied from various perspectives. Consequently, traditional AI research has evolved into many subfields, each has its problems and proposed solutions, such as search, reasoning, learning, planning, perception, action, etc. To build an AGI, it is very natural to adopt an *integrative approach*, by combining the most premising solution in each subfield to cover all aspects of intelligence.

Another justification of the integrative approach is that since every AI techniques has its strength and weakness, to use all of them together should give us the best solution. Consequently, people have attempted to integrate multiple techniques, like rule-based, case-based, statistical, connectionist, evolutionary, robotic, etc., into a hybrid system, in which each technique is used for what it is best for.

The above opinions are all reasonable, and research projects on integrative AGI have produced interesting results, as described in several chapters of this volume. However, it is not the only possibility.

NARS does not take an *integrative* approach, but a *unified* approach. Though its design has been influenced by the ideas of various schools in AI and Cognitive Sciences, the resulting technique is a singleton, rather than a "toolbox" consists of multiple independently developed tools.

It is possible for NARS to follow a unified approach, first because of its aim. As discussed previously, in this research "intelligence" is seen as a principle, not a group of loosely related capabilities or functions. It is much more likely to use a single technique to realize a principle, than to realize multiple independent capabilities.

Furthermore, the unified approach is adopted in NARS, not because it is easier than the integrative approach, but because it has theoretical and practical advantages.

Scientific research always prefers unified explanation of complicated phenomena. Driven by similar motivations, many previous AI projects attempted to use a single technique to cover most of the field, if not all of it. General Problem Solver tried to treat all problem-solving processes as heuristic search in a state space [8]; the Fifth Generation Computer System was based on the assumption that many AI problems can be solved by parallel inference [9]. Their failures make many people to believe that AGI cannot be achieved by a single technique, given its well-known complexity. However, this kind of evidence is far from conclusive. The previous failures may be caused by their selections of aims and/or techniques, and they cannot rule out the possibility that when the aim of the research is properly set, certain technique can achieve it. Also, a system based on a single technique can still produce very complicated behavior, given the complexity of the environment of the system.

From an engineering point of view, the most difficult part of the integrative approach is not in the building of individual "modules" using different techniques, but in organizing them into a whole [10]. To passing data from module to module is easy, but it is hard to interpret the data in a wide range of situations consistently in different modules, because the involved techniques have been developed on very different theoretical foundations. When an integrative/hybrid system does not work properly, it is not easy to say what is wrong, because the lack of a common basis of the system. On

the contrary, the development and evaluation of a unified system is much better guided by the theory behind the technique. Even if it eventually fails to cover the whole field, we can still get a much more clear picture about the capability and limitation of the technique, compared to the situation where the technique is just one of several in the system.

Since NARS can use various techniques as tools for practical problems, it will not attempt to directly use a single technique on all kinds of problems. Instead, the heterogeneous tools will be handled in a unified manner. For a concrete problem, the system will prefer to apply an available tool, and it is when no such a tool is there or can be easily made, will the system try to solve it using the reasoning ability.

Finally, even though human intelligence can be studied from different perspective, there is still evidence suggesting the close relationship of the various facets. As Piaget commented, intelligence "appears as a total system of which one cannot conceive one part without bring in all of it" [11].

In summary, now it is still too early to decide whether AGI is better pursued by an integrative approach or a unified approach. What is argued above is that the unified approach is not as obviously wrong as many people currently believe.

4.3 Why to work in a reasoning system

Even among people who agree that AGI may be achieved using a single technique, reasoning system, or logic-based system, is still not favorable, for various reasons [12]. Typical objections include:

- "Reasoning must be applied on materials provided by a sensorimotor mechanism, which was also evolved before high-level cognition."
- "Logic is too abstract, while real intelligence should be grounded, situated, and embodied."
- "Logic is too rigid to capture the flexibility of intelligent behaviors, which are often irrational."
- "Reasoning technique has been studied in AI for decades, and the results are not promising."

These opinions are all acceptable, as far as the involved "logic" refers to first-order predicate logic or its variants, such as non-monotonic logics. However, people often forget that in its original and broad sense, "logic" is just the attempt of capturing valid patterns of inference in a content-independent manner, and "inference" is just the process by while new knowledge is derived from existing knowledge. First-order predicate logic was originally designed to provide a logic foundation *for mathematics*, so if it cannot be used to capture the patterns of human thinking in general, it still does not mean that no logic can do that.

The logic implemented in NARS, Non-Axiomatic Logic, is fundamentally different from traditional mathematical logic, in that it is an attempt to capture the principle of adaptation with insufficient knowledge and resources. In this logic, a "term" is an identifiable item or pattern in the system's experience; a "statement" is an relation between two terms indicating their substitutability; the "truth-value" of a statement measures how the statement is supported or refuted by the system's experience; the "meaning" of a term indicates the role it plays in the system's experience; the function of an "inference rule" is to accomplish a single inference step, which build term(s) and/or statement(s) to summarize the information in existing ones; and a "reasoning process" is a sequence of step carrying out the tasks in the system for surviving and adapting. Since these notions are used in such broad senses, in NARS

"reasoning" covers many activities that are usually called by other names, such as "learning", "planning", "perceiving", and so on.

Such a system is situated and embodied, because truth and meaning are grounded in the system's experience, and sensitive to the change of context. The behaviors of the system are "rational" with respect to its available knowledge and resources, though there is no guarantee that all of its predictions will be confirmed by its future experience, or to be judged by an observer as "rational" according to knowledge and resources that are unavailable to the system.

The reasoning technique used in NARS not only solves or avoids many problems in traditional logic-based AI systems [1], but also has many advantages over competing techniques (such as problem-specific algorithms, Bayesian networks, connectionist models, genetic algorithms, and reactive robots):

- It uses a domain-independent language for knowledge representation, which is more expressive than problem-specific data structures or numerical vectors in coding items and patterns in the system's experience.
- It uses an experience-grounded semantics to associate the terms and statements to the items and patterns in the (symbolic and/or sensorimotor) experience of the system.
- It uses inference rules to regulate individual steps of its working process, and each rule is justifiable in a domain-independent manner, according to the same semantics.
- It allows the system to organize the individual steps into task-processing courses at run time, so as to achieve flexibility and creativity. Consequently, the system can handle novel tasks by the cooperation of reliable basic actions, according to experience and context.

Though building a reasoning system is not necessarily the only way to AGI, there are reasons to say that it seems more promising than the other proposed alternatives.

4.4 Why to have these stages and augmentations

As described earlier, the development if NARS can be roughly divided into four stages, each of which is built on the top of the previous ones. After completion, NARS can be further augmented in (at least) six different directions, which are largely independent of each other.

This plan makes the development of the system *incremental*, in the sense that in each phase of the project, the immediate aim is relatively simple and clear, so it is possible to be carried out with a small expense. After each phase, the system is "more intelligent", in terms of its ability of adapting to its environment and its efficiency of using its knowledge and resources.

Though this incremental development approach builds the system one part as a time, it is not "integrative", since each stage is not built in isolation, but on top of the previous parts. Each of the six augmentations is indeed developed independent of each other, but is still based on NARS as a whole.

On the other hand, this dependency is one-way, since an earlier stage in NARS does not depend on any later stage, and the design of NARS as a whole does not depend on any of the augmentations. As discussed previously, though each augmentation makes the system more intelligent, and has important practical application, it is optional for an AGI system. On the contrary, NARS is the common core of any AGI system developed by following this research plan.

Almost for every augmentation listed before, there are people who believe either that it is more crucial than the ability of reasoning for an AGI, or that it should be taken as a necessary part of any AGI, together with reasoning. If we analyze each of these beliefs in detail, we will see that it is usually based on a conception of intelligence that identifies it with certain human behavior or capability. As stated before, such a conception tends to define intelligence too close to its only known example, human intelligence, to explore the possible forms of intelligence.

In that case, why the ability of reasoning is absolutely necessary for an AGI? As discussed above, in NARS the notion of "reasoning" is extended to represent a system's ability to predict the future according to the past, and to satisfy the unlimited resources demands using the limited resources supply, by flexibly combining justifiable micro steps into macro behaviors in a domain-independent manner. There are reasons to believe that any AGI needs this ability.

Because of this, NARS is not merely a reasoning system, but an attempt to model the core of any thinking machine in an abstract manner. To satisfy additional requirements, we may also plug various sensors and effecters into the machine, teach it natural languages, train it with commonsense and domain-specific knowledge, raise it in a multi-system community, run it in specially-designed hardware, or let it evolve from generation to generation. Even so, we still need to build the core first.

Acknowledgement

The author thanks the anonymous reviewer for helpful comments.

Reference

[1] P. Wang, Rigid Flexibility: The Logic of Intelligence, Springer, 2006.
[2] P. Wang, On the Working Definition of Intelligence, Technical Report No. 94, Center for Research on Concepts and Cognition, Indiana University, 1994.
[3] L. Barsalou, Perceptual symbol systems, Behavioral and Brain Sciences 22 (1999), 577-609.
[4] R. Brooks, Intelligence without representation, Artificial Intelligence 47 (1991), 139-159.
[5] A. M. Turing, Computing machinery and intelligence, Mind LIX (1950), 433-460.
[6] D. Lenat and E. Feigenbaum, On the thresholds of knowledge, Artificial Intelligence 47 (1991), 185-250.
[7] J. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, MIT Press, 1992.
[8] A. Newell and H. A. Simon, GPS, a program that simulates human thought, E. A. Feigenbaum and J. Feldman (editors), Computers and Thought, 279-293, McGraw-Hill, 1963.
[9] E. Feigenbaum and P. McCorduck, The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World, Addison-Wesley, 1983.
[10] A. Roland and P. Shiman, Strategic Computing: DARPA and the Quest for Machine Intelligence, 1983-1993, MIT Press, 2002.
[11] J. Piaget, The Origins of Intelligence in Children, W.W. Norton, 1963.
[12] L. Birnbaum, Rigor mortis: a response to Nilsson's "Logic and artificial intelligence", Artificial Intelligence 47 (1991), 57-77.