PAPER

Associative Semantic Memory Capable of Fast Inference on Conceptual Hierarchies

Qing Ma† *and* **Hitoshi Isahara**†, *Members*

SUMMARY The adaptive associative memory proposed by Ma is used to construct a new model of semantic network, referred to as associative semantic memory (ASM). The main novelty is its computational effectiveness which is an important issue in knowledge representation; the ASM can do inference based on large conceptual hierarchies extremely fast—in time that does not increase with the size of conceptual hierarchies. This performance cannot be realized by any existing systems. In addition, ASM has a simple and easily understandable architecture and is flexible in the sense that modifying knowledge can easily be done using one-shot relearning and the generalization of knowledge is a basic system property. Theoretical analyses are given in general case to guarantee that ASM can flawlessly infer via pattern segmentation and recovery which are the two basic functions that the adaptive associative memory has.

key words: adaptive associative memory, pattern segmentation and recovery, conceptual hierarchy, inference, computational effectiveness

1. Introduction

If we analyze human behavior we find that in spite of operating with a large knowledge base, human agents take but a few hundred milliseconds to perform a broad range of cognitive tasks, such as recognizing objects, understanding spoken and written text, and making inferences such as: "Tweety is a bird therefore it flies". The human performance data indicates that the representation of conceptual information and cognitive processes that access it are computationally effective. The question of computational effectiveness therefore must be tackled at the very offset for understanding the principles underlying the organization and use of conceptual information in cognitive systems [20][21]. The traditional AI approaches (i.e., symbolic methods), however, cannot attain such a computational effectiveness. To address this important issue in knowledge representation, we think that the artificial neural networks, among which particularly the associative memories, are the appropriate paradigm.

Associative memories [7] are among the most extensively studied and best-understood paradigms in the field of artificial neural networks because of their human-like manner of information storage and retrieval. The basic attribute of conventional associative memories proposed so far is one-to-one associative function in heteroassociative types or pattern recovery function in autoassociative types. Several studies have shown that when an associative memory is equipped with one-to-many associative function (or pattern segmentation function in autoassociative types), it becomes a powerful model to address various cognitive processes [8]-[10],[13]. It has been particularly indicated by these studies that the procedural knowledge, a kind of knowledge consisting of the skills of perceptual motion or cognition, can be well represented by such an associative memory. This paper further demonstrates that the semantic knowledge, another kind of knowledge for understanding spoken and written text, making inferences, and so on can also be well represented by such an associative memory. Thus, both this two kinds of knowledge can be addressed within the same framework of the associative memory. The associative memory can therefore be regarded as an essential module of cognitive systems in hardware level. According to Marr's theory [17], it is indispensable to understand cognitive systems in hardware level, i.e., in the details of how the computational theory or algorithm is implemented.

Semantic networks (accurately conceptual hierarchies) express semantic knowledge in terms of concepts, their properties, and the hierarchical relationships between concepts. Since their introduction by Quillian [18], semantic networks have played a significant role in knowledge representation and form the backbone of many cognitive and AI systems [6]. To date, several connectionist models of semantic networks (e.g., [4][5][11][20][21][23]) have been proposed to attain computational effectiveness; they could do inference in time that is logarithmic in the number of concepts in the knowledge base. As represented by Shastri's system [20], most of them used controlled spreading activation as a computational primitive and have very complex architectures. A few models have been proposed that use typical artificial neural networks proposed so far (e.g., associative memory and the Boltzmann machine). These models usually have simple and easily understandable architectures, but lack sufficient ability to handle general problem cases.

The Hinton's network [11], the one most closely related to our model, for example, uses a kind of as-

Manuscript received *******

Manuscript revised ********

†The authors are with the Communications Research Laboratory, Ministry of Posts and Telecommunications, 588-2, Iwaoka, Nishi-ku, Kobe, 651-24, Japan

sociative memory in which higher-order problems, inheritance and recognition inference based on conceptual hierarchies, are attempted to be solved by reducing them into lower-level processing, i.e., pattern recovery. More specifically, the network encodes conceptual hierarchies by a set of triples of the form: (concept, property, value). Given two components of a triple, the network can determine the third tuple. However, because the network performs inference only by pattern recovery, there are inherent problems. First, if there are exceptions in the conceptual hierarchies, i.e., if there are concepts that do not inherit properties from their ancestors (such exceptions always exist in the real world), interference may be caused by these exceptions. As a result, inference takes longer and the accuracy drops. Evidently, if conceptual hierarchies are large (so that there are more exceptions), the interference may become so strong that inference cannot be performed correctly. Furthermore, it is almost impossible to obtain correct answers for complex recognition inference problems that are given by a large number of clues. Besides these inherent problems, the network also suffers heavily from the shortcomings of the perceptron algorithm that it uses. For example, in large conceptual hierarchies, learning takes a very long time, and convergence cannot be guaranteed. Moreover, new knowledge can only be added by relearning the entire conceptual hierarchy, and unnecessary and/or incorrect knowledge cannot be deleted.

Our proposed model resolves the dilemma of obtaining high performance while at the same time maintaining a simple and easily understandable architecture. For its kernel, it uses the adaptive associative memory (AAM) [16] proposed by Ma, which can do pattern segmentation as well as pattern recovery. We therefore refer to the proposed model as the associative semantic memory (or ASM for shortly). ASM performs inference much faster than existing models; the computation time tends not to increase with the size of the conceptual hierarchy. By using pattern segmentation, ASM resolves the problems in Hinton's network: it solves inference problems with exceptions and complex recognition problems flawlessly. Because one-shot learning is used in ASM, there are also no learning problems, the same as in Hinton's network, and the modification (addition and deletion) of knowledge is very easy. As in Hinton's network, generalization is a basic property of ASM in the sense that concepts automatically acquire any effects possessed by their subordinate concepts.

In this paper we begin by describing conceptual hierarchies, a class of inference problems based on them, and how the inference problems can be formalized and transformed into lower level processing, i.e., pattern segmentation and pattern recovery. We then introduce AAM, the kernel of ASM. Next we describe ASM and theoretically show that ASM performs inference perfectly. We also show that the inference time does not increase with the size of the conceptual hierarchy. We explain why knowledge can be easily modified and that the generalization of knowledge is a basic system property. We next conduct several computer simulations to confirm our theoretical analyses and to evaluate how fast ASM performs inferences. Finally, we conclude this work and point out the future extension of this research.

2. Conceptual hierarchies

Concepts are primitives of semantic knowledge in cognitive science [12][19][22] and each concept can be characterized by a set of features which are also instances of concepts. For example, the concept "dog" can be characterized by a set of features (*mammal, fur, teeth, four legs,* $\cdots$,). Semantic knowledge can therefore be represented by a set of such concepts which are related with each other via their properties, e.g., is_a, part_of, and so on. If we use nodes to express concepts and labeled links to express properties (the relationships between concepts), semantic knowledge can be represented by a graph which consists of nodes and links. Such graphs are called semantic networks.

The semantic networks specially from the conceptual hierarchical point of view are called conceptual hierarchies. As pointed out by Shastri [20], the characterization of conceptual hierarchies is broad enough to capture the basic organizational principles underlying frame-based representation languages such as KRL [1] and KL-ONE [2]. In real world, we can find many examples of conceptual hierarchies. Roget's thesaurus [3] constructed from the viewpoint of linguistics and cognition, for example, is a very large word (conceptual) hierarchy in which more than 325,000 words and phrases are divided into six fields and the words in each field are further classified into more than 1,000 meaning categories. This kind of dictionaries has been constructed in several languages and widely used in explaining human language activities (phenomena) and constructing natural language processing systems.

2.1 A class of inference: inheritance and recognition

Figure 1 shows an example of conceptual hierarchies. A basic characteristic in conceptual hierarchies is property inheritance: the properties of superordinate concepts are inherited by their subordinate ones. Property inheritance leads to the efficient storage of knowledge. However, a class of basic inference, inheritance and recognition, based on conceptual hierarchies is required to obtain needed knowledge.

Inheritance inference enables us to infer the properties of concepts based on those of their ancestors. Suppose, for example, we know elephants are a mammal and want to know how many legs elephants have. Symbolically, we find the node for *Elephant*. If there is no information about legs attached directly to this node,

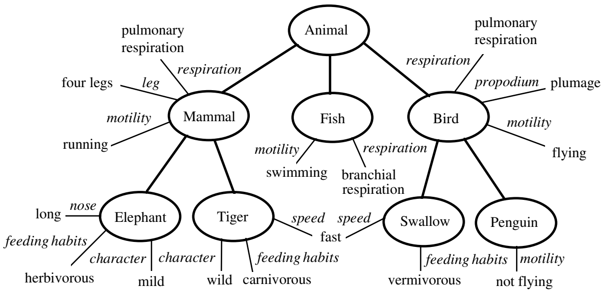

Fig. 1 Real-world example of conceptual hierarchies.

then the node for *Elephant*'s ancestor, *Mammal*, is checked. If the answer is again not attached to the node, then its ancestor, *Animal*, is examined, and so on. In Fig. 1, the answer can be found in the *Mammal* node. Consequently, to answer a seemingly simple query with a symbolic approach, we might have to visit many nodes in the data base to retrieve the desired information.

Recognition inference enables us to find individual concepts that have multiple specific properties. When we are told that something has four legs, runs fast, and so on, for example, we can use the conceptual hierarchy shown in Fig. 1 to infer that it is a tiger. Symbolically, the sets {*Mammal, Elephant, Tiger*} with *four legs* and {*Tiger, Swallow*} with *runs fast* are found first. Then, the answer is obtained by examining the intersection of these sets. In this example, *Tiger* is obtained. Although *Tiger* has both *four legs* and *runs fast*, these properties are not necessarily available locally at the node for *Tiger*. They may have to be determined via inheritance inference. Doing set-intersection operations symbolically takes much longer time compared to doing inheritance inference.

In the real world there are exceptions to property inheritance. For example, birds are thought to fly, but some, such as penguins, do not. Exceptions in conceptual hierarchies give rise to nonmonotonicity and ambiguity, neither of which can be handled using first-order logic.

2.2 Formalization of inference problems

Conceptual hierarchies include two kinds of knowledge: hierarchical relationships between concepts (e.g., *Elephants are mammals*), which can be represented using a special encoding procedure that will be described in Sec. 4, and knowledge about each concept, which can be represented by triples $(concept, property, value)$ (e.g., *(Elephant, feeding habits, herbivorous)*).

By using triple expressions, any inheritance inference problem can be formalized as (*concept: c, property: p, value: ?*), where *concept: c* and *property: p* are clues and *value: ?* is the solution (*value: v*) that should be found to fill the triple so that the knowledge represented by this triple is true (i.e., in agreement with the conceptual hierarchy). Note that triple (*concept: c, property: p, value: v*) is not necessarily found locally at the representation of *concept: c*. For example, for the triple (*Elephant, respiration, pulmonary respiration*), the property *respiration* and its value *pulmonary respiration* are not directly attached in the representation of *Elephant* as shown in Fig. 1. In connectionist representation, each element of a triple is encoded as a binary pattern, which can be regarded as a one-dimensional vector whose elements (bits) have value zero or one. The inheritance problem can thus be rewritten as $(pattern(c), pattern(p), pattern(\#))$ or more simply, $pattern(c, p, \#)$, where $pattern(\#)$ is the deficient part to be recalled, using pattern recovery (and pattern segmentation as the case may be; see Sec. 4 for more detail).

Also, any recognition inference problem can be formalized as (*concept: ?, property:* p_1*, value:* v_1) & (*concept: ?, property:* p_2*, value:* v_2) & $\cdots$ & (*concept: ?, property:* p_m*, value:* v_m), where *property* : p_i and *value* : v_i $(i = 1, \cdots, m)$ are clues and *concept: ?* is the solution (*concept: c*) that should be found to fill these triples so that all the knowledge represented by these triples is true. Note that (*concept: c, property:* p_i*, value:* v_i) $(i = 1, \cdots, m)$ also need not necessarily be found locally at the representation of *concept c*. In connectionist representation, the recognition problem can further be rewritten as $(pattern(\#), \sum_{i=1}^{m} pattern(p_i), \sum_{i=1}^{m} pattern(v_i))$ or more simply, $pattern(\#, \sum_{i=1}^{m} p_i, \sum_{i=1}^{m} v_i)$, where $\sum_{i=1}^{m} pattern(x_i)$ $(= pattern(\sum_{i=1}^{m} x_i))$ is a composite (superimposed) pattern composed of $pattern(x_i)$ $(i = 1, \cdots, m)$. Here, composite patterns are also binary ones (i.e., they are also one-dimensional vectors whose elements (bits) have value zero or one) that are obtained by adding all bits with the same suffix number in the multiple patterns in a Boolean way (a logical addition). The recognition problem can therefore be regarded as a composite pattern with deficient part. To recall the deficient part, the composite pattern must be separated and recovered into a single pattern like $pattern(c, p_1, v_1)$.

3. Adaptive Associative Memory (AAM)

Adaptive associative memory (AAM) [16] can separate and recover composite patterns which might be degraded by deficiency or noise. An example where AAM separates and recovers a deficient composite input is shown in Fig. 2, in which the black dots indicate the bits that have positive values and the others indicate the bits that have value zero. Since such patterns can be easily transformed into binary patterns by using a binary thresholding function, which will be shown later (see Eq. (4)), the bits shown by black dots are called "1-bits" and the others are called "0-bits" in case with no confusion.

Figure 3 shows a simplified structure of AAM com-

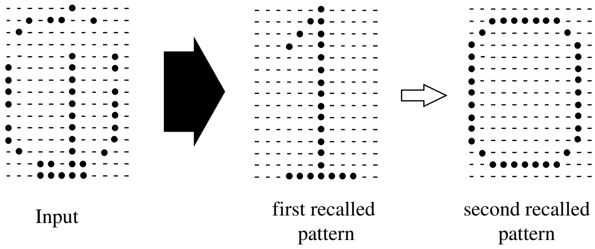

Fig. 2 Example of recall from a deficient composite input.

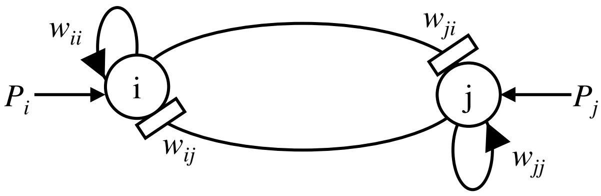

Fig. 3 Simplified structure of AAM composed of two units.

posed of two units. The units are inhibited with each other via inhibitory connections w_{ij}/w_{ji} and have positive self-loops via auto-connections w_{ii}/w_{jj}. AAM consists of a fully-interconnected set of such units. It operates synchronously and the dynamics can be expressed as

$$s_i(k+1) = \varphi(w_{ii}(k)s_i(k) - \sum_{j \neq i} w_{ji}s_j(k) + cP_i + b), \quad (1)$$

where $s_i(k)$ denotes the activity of unit i at time k, P_i is the external input (0 or 1), c and b are, respectively, the connecting coefficient from P_i to unit i and the bias of unit i, and $\varphi(x)$ is an analog thresholding function:

$$\varphi(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leqq 0. \end{cases} \quad (2)$$

The activities of units in AAM are reset when a new input is given; i.e., $s_i(0) = 0$.

As mentioned in Sec. 2.2, the patterns shown in Fig. 2 can be regarded as one-dimensional vectors. Their *ith* element (bit) corresponds to unit i of AAM and the value of the *ith* bit equals to the activity $s_i(k)$ at stable state. Since the basic function of AAM is the pattern segmentation, a positive bias b in (1) has to be introduced into AAM to enable pattern recovery. By setting a positive bias b, all units of AAM are initially with positive activities. When a single pattern (e.g., "0" as shown in Fig. 2) with some deficient 1-bits is given, for the 1-bits that exist, the corresponding units of AAM will initially have large positive activity $c + b$ (see Eq. (1)) and for all other bits including the 1-bits that are deficient, the corresponding units of AAM will initially have small but non-zero (positive) values b. That is, we can equivalently regard that all bits of the pattern including the part to be recovered and its background are non-zero, but the bias b of AAM is zero instead. To recover a pattern (e.g., "0") is thus performed by separating it from its background. As a result, the complete pattern (e.g., "0"), in which the 1-bits that originally exist have large positive values $c+b$, the deficient 1-bits have small positive values b, and the remaining bits with value zero, can be obtained. On the other hand, if bias b is set to zero, the units of AAM corresponding to the 1-bits that are deficient are always with activity zero and the deficient "1-bits" cannot be recovered.

The auto-connection weight $w_{ii}(k)$ changes adaptively to reduce the recall of spurious memories and to speed up the recall time (see details in Ref. [16]). Specifically, it changes according to the activities of the other units j ($j \neq i$) that are weighted with the corresponding inhibitory connections:

$$w_{ii}(k) = \alpha \sum_{j \neq i} w_{ji}\psi(s_j(k)), \quad (3)$$

where α is a positive coefficient called adaptive weight and $\psi(x)$ is a binary thresholding function:

$$\psi(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leqq 0. \end{cases} \quad (4)$$

Since the activities of units in AAM are reset when a new input is given, the auto-connection weight $w_{ii}(0)$ is therefore also reset by (3).

The weight of inhibitory connection w_{ji} is decided at the learning phase according to the following one-shot learning algorithm:

(i) Initially, $w_{ji} = W$ for all i and j ($i \neq j$), where $W > 0$
(ii) When a learning pattern is presented,

$$w_{ji} = \begin{cases} 0 & \text{if } P_i \cdot P_j > 0 \\ w_{ji} & \text{if } P_i \cdot P_j = 0. \end{cases} \quad (L-1)$$

From $(L-1)$ we can see that inhibitory connections are symmetric; i.e., $w_{ij} = w_{ji}$. After the learning phase, the autocorrelations of the learned patterns are formed by the inhibitory connections that have disappeared. In other words, the inhibitions between the 1-bits in the same learned pattern are removed, but those between 1-bits in different learned patterns remain intact. Thus, if a deficient composite pattern (see Fig. 2 for example) is input, the patterns constituting the input ("0" and "1" in this example) compete for survival via the intact inhibitory connections, w_{ji}. One pattern ("1" in this example) finally wins and is recalled. More strictly, we have the following corollary derived in [16].

Corollary 1: (a) When a single stored (learned) pattern is input, the input itself appears as a stable state; i.e., the input itself is recalled. (b) When a composite pattern is input, an AAM can reach a stable state (recall a pattern) in which there are no inhibitions among the 1-bits.

Actually, the recalled pattern described in *Corollary 1* is most probably a single stored pattern; i.e., spurious memories are rarely recalled. The probability that no spurious memories are recalled is 1 by the following corollary which was also derived in [16].

Corollary 2: When M patterns, $P^{(X_1)}$, $P^{(X_2)}$, $\cdots$, $P^{(X_M)}$, are stored, for any composite input, a sufficient condition to guarantee that the desired patterns are recalled (i.e., no spurious memories are recalled) is as follows:

$$\text{for } P^{(A)} \in \{P^{(X_1)}, P^{(X_2)}, \cdots, P^{(X_M)}\},$$

$$\exists i \in P^{(A)} \quad \text{such that} \prod_{j\in\ P^{(B)}} w_{ij} \neq 0. \qquad (C-1)$$

Here, a pattern P is regarded as a set of suffix numbers (e.g., i and j) whose bits are with value one and $P^{(B)} = \bigcup_{l=1}^{M} P^{(X_l)} - P^{(A)}$. This $(C-1)$ means that a 1-bit i should exist in each pattern $P^{(A)}$ such that all the inhibitory connections between the 1-bit i and the 1-bits of any other stored pattern are non-zero after learning.

By suppressing the competitive strength of the currently recalled pattern ("1" in the example shown in Fig. 2), the next pattern ("0" in this example) can be recalled. Suppression is done by a readout control unit, RCU_i, attached to each unit of AAM (not shown in Fig. 3, but shown in Ref. [16]). Note that once a pattern is recalled, the same pattern will not be recalled again.

In competitive process, because each unit has a positive self-loop (the first terms in (1) and (3)), the more inhibitions received by an AAM unit, the stronger the unit will become. We can thus easily obtain the following corollary, which is closely related to the present work.

Corollary 3: Supposing that all the stored patterns satisfy $(C-1)$, among the stored patterns in competition, the one that has a 1-bit receiving the most inhibitions from other patterns will win the competition (be recalled first).

It should be noted that AAM itself has high performance for pattern segmentation and recovery even if $(C-1)$ is not satisfied. However, to construct a model of semantic network that can perform inference perfectly, the stored patterns must be encoded to satisfy $(C-1)$.

4. Associative semantic memory (ASM)

4.1 Architecture

Our proposed ASM (Fig. 4) comprises an encoder, a decoder, and a single adaptive associative memory (AAM), which is divided into three parts: CON, PROP, and VAL. The states of these three parts are used to represent concept, property, and value of triples. Within CON, we assume that there are no inhibitory connections between units. Besides this constraint, the units in the three parts are connected with each other: not only within PROP and VAL, but also between all three parts. As shown in Fig. 4, the output (stable state) of AAM (P_{OPT}=$pattern(c_o, p_o, v_o)$) is fed back to the encoder; $pattern(c_o)$, $pattern(p_o)$, and $pattern(v_o)$ are the stable states of CON, PROP, and VAL, respectively.

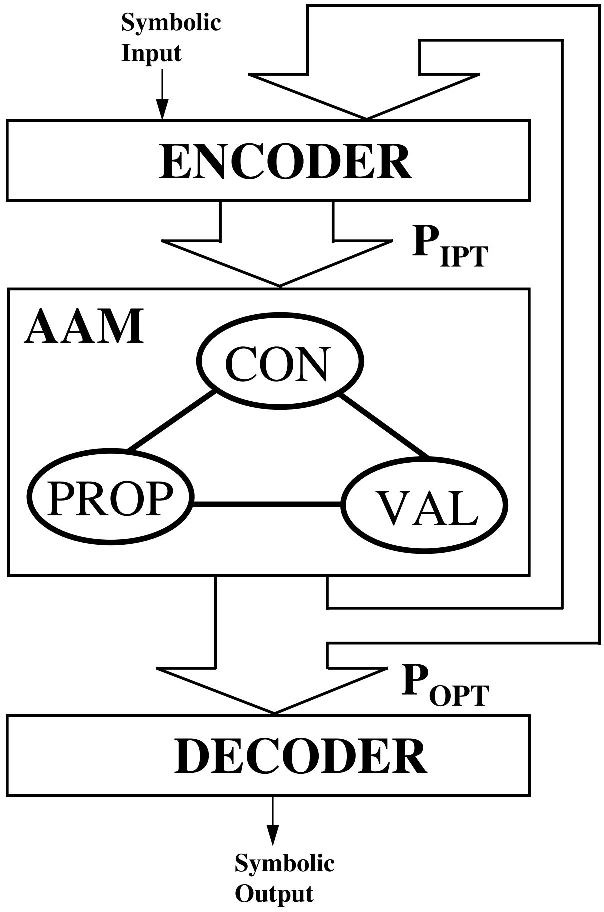

Fig. 4 Structure of ASM.

The encoder initially transforms symbolic input like $(c, p, ?)$ into spatial pattern $P_I = pattern(c, p, \#)$. Then, it further combines P_I with the output of AAM, if it exists, to form an input to AAM, i.e., $P_{IPT} = P_I + P_{OPT} = pattern(c + c_o, p + p_o, v_o)$. Thus, $pattern(c + c_o)$, $pattern(p + p_o)$, and $pattern(v_o)$ are given to CON, PROP, and VAL, respectively. In the same way, for recognition problem (*?*, p_1, v_1) & (*?*, p_2, v_2) & $\cdots$ & (*?*, p_m, v_m), the input to AAM will be $P_{IPT} = pattern(c_o, \sum_{i=1}^{m} p_i + p_o, \sum_{i=1}^{m} v_i + v_o)$. Note that before AAM first reaches a stable state, there is no output from AAM, i.e., $P_{OPT} = pattern(c_o, p_o, v_o) = \#$. On the other hand, the decoder extracts subpattern $pattern(v_o)$ or $pattern(c_o)$ corresponding to the kind of problems (inheritance or recognition inference) from output P_{OPT} of AAM and transforms it into symbolics representing the solution. Wishfully, the encoder and decoder are expected to be automatically constructed in the sense of self-finding the rules for encoding concepts in hierarchy, using self-organization techniques. In this paper, however, they are both assumed to exist and their modelings are left as a future work.

ASM has been designed to give an answer through at most two stages (i.e., AAM will at most do two sequential recalls or twice reach the stable state). At the first stage, bias b of the AAM in Eq. (1) is set to a positive value so that both pattern recovery and pattern segmentation are possible. The input to AAM is simply the pattern transformed from the symbolic input, i.e., $P_{IPT} = P_I$. For this input, AAM will reach stable state P_{OPT1} through several steps. If $P_{OPT1} \supseteqq P_{IPT}$, ASM will not to proceed to the second stage and decode P_{OPT1} to obtain a symbolic answer. Otherwise, ASM will proceed to the second stage. The input to AAM is the pattern composed of the pattern transformed from the symbolic input and the output of AAM obtained at the first stage, i.e., $P_{IPT} = P_I + P_{OPT1}$. Bias b of the AAM in Eq. (1) is set to 0 so that only pattern segmentation is possible, i.e., only the subpattern of P_{IPT} can be recalled. Thus, at the second stage, AAM will reach another stable state, P_{OPT2}, which will be decoded to obtain a final answer. Note that P_{OPT1} and P_{OPT2} are sequentially recalled patterns, and P_{OPT2} is therefore different than P_{OPT1} (see details in [16]).

4.2 Encoding and storing of conceptual hierarchies

To make ASM behave as expected, special encoding for conceptual hierarchies is needed. To represent hierarchical relationships between concepts, similar to Hinton's network, the concepts in the hierarchical relationships are encoded as patterns with inclusion relations. The concepts in the same levels (i.e., the concepts rooted in one ancestor) are encoded into patterns that satisfy $(C-1)$ and have the same number of 1-bits so that whichever pattern wins the competition first is not affected by the structure of the pattern itself. If we consider the conceptual hierarchy in Fig. 1, for example, *Elephant* may be encoded in $pattern(Elephant) = (1,0,0,0)$, $Tiger$ may be encoded in $pattern(Tiger) = (0,1,0,0)$, and their ancestor, $Mammal$, may therefore be encoded in $pattern(Mammal) = (1,1,0,0)$, which consists of both $pattern(Elephant)$ and $pattern(Tiger)$. All properties and values are also respectively encoded into patterns that satisfy $(C-1)$, and the numbers for the 1-bits in these patterns are set the same; a simplest way is to, respectively, use one 1-bit to represent one property and value. Note that because the concepts, properties, and values are encoded independently, the encodings for the three elements are not affected with each other and it is not difficult to encode a large-scale conceptual hierarchy. Needless to say, these special encodings destroyed the real connectionist sense. By introducing such symbolic encoding in a connectionist system, however, the inference can be performed perfectly and the time required does not increase with the size of conceptual hierarchy. This cannot be done by purely symbolic methods or previous connectionist systems.

By using such encoding methods, a conceptual hierarchy can be stored in ASM by simply learning a set of patterns for triples (c_i, p_{ij}, v_{ij}), in which (p_{ij}, v_{ij}) is the jth pair of a property and its value attached to the node for the ith concept c_i (e.g., (*Elephant, feeding habits, herbivorous*) in Fig. 1). By assuming no connections among units in CON and the special encoding described above, it can easily be known that for any stored pattern $pattern(c,p,v)$, the 1-bits in $pattern(p)$ and $pattern(v)$ will always have a larger, or at least the same, number of inhibitory connections (with the 1-bits in other stored patterns) than the 1-bits in $pattern(c)$. Thus, from *Corollary 3* we can further obtain the following corollary.

Corollary 4: When l patterns $P_i = pattern(c_i, p_i, v_i)$ $(1 \leqq i \leqq l)$ stored in ASM are in competition, pattern P_h $(1 \leqq h \leqq l)$ will be recalled first if and only if its subpattern $pattern(p_h)$ or $pattern(v_h)$ has a 1-bit that receives the most inhibitions compared to the other $l-1$ subpatterns, $pattern(p_i)$ and $pattern(v_i)$ $(i \neq h)$, without taking account of $pattern(c_i)$ $(1 \leqq i \leqq l)$.

4.3 Inheritance inference

In this subsection we use a general analysis case to show that ASM can correctly perform inheritance inference via pattern segmentation and recovery. The inference problems handled in this and the next subsections are assumed to be solvable using ASM's knowledge, i.e., the answer can be uniquely determined from input by using the conceptual hierarchy stored in ASM. The *unknown*, *clashing*, and *ambiguous* problems will be described in Sec. 4.5.

For inheritance problem $(c,p,?)$ we suppose v to be the correct solution. At the first stage, the pattern input to AAM is $P_{IPT} = P_I = pattern(c,p,\#)$, and both pattern recovery and pattern segmentation are possible. When solution v is not an exception to property inheritance, there is a pattern [shown in (5)] stored in ASM that includes input pattern P_{IPT}:

$$pattern(c,p,v) \text{ or } pattern(c_0,p,v), \tag{5}$$

where c_0 is c's ancestor, i.e., $pattern(c_0) \supseteqq pattern(c)$. On the other hand, when solution v is an exception to property inheritance, there are two patterns [shown in (6)] stored in ASM that include input pattern P_{IPT}:

$$pattern(c,p,v) \text{ and } pattern(c_0,p,v_0), \tag{6}$$

where c_0 is c's ancestor and $v \neq v_0$. In general, we can assume that in the conceptual hierarchy there are further m concepts, $c_1, \cdots, c_m$ ($m \geqq 0$), that satisfy the following conditions:

- they are c_0's ancestors, i.e., $pattern(c_i) \supseteqq pattern(c_0)$ for $i = 1, \cdots, m$;
- they are in hierarchical relationships, i.e., $pattern(c_i) \subseteqq pattern(c_j)$ for $i < j$ and $i, j = 1, \cdots, m$;
- they have different values v_i in property p, i.e., all v_i ($i = 0, \cdots, m-1$) are exceptions in property inheritance.

Thus, there are additional m patterns [shown in (7)] also stored in ASM that include input pattern P_{IPT}:

$$pattern(c_i, p, v_i),\ i = 1, \cdots, m. \tag{7}$$

The patterns in (5) and (7), or in (6) and (7), compete with each other when $pattern(c, p, \#)$ is given.

Lemma 1: When the patterns in (5) and (7), or in (6) and (7), compete with each other, in both cases the pattern that has value v, i.e., $pattern(c, p, v)$ or $pattern(c_0, p, v)$ in (5), or $pattern(c, p, v)$ in (6), is recalled first.

Proof: Appendix A.

The correct solution v can thus be obtained. Because the recalled pattern of the triple includes the input pattern, ASM stops processing at the first stage.

Let us assume the conceptual hierarchy shown in Fig. 1 is stored in ASM. For the (*Tiger*, *motility*, ?) problem, i.e., $P_{IPT} = (Tiger, motility, \#)$, for example, because the only pattern stored in ASM that includes the input pattern is $pattern(Mammal,$ $motility,\ running)$, this pattern can be regarded as $pattern(c_0, p, v)$ in (5). Thus, this pattern is recalled first by *Lemma 1*. That is, $P_{OPT1} = pattern(Mammal, motility, running)$. Because $P_{OPT1} \supseteqq P_{IPT}$, which means that the input pattern has been recovered perfectly, it is therefore not needed to proceed to the second stage. Thus, v (=*running*) can be obtained as the solution. On the other hand, for the problem $(Penguin, motility, ?)$, i.e., $P_{IPT} = pattern(Penguin, motility, \#)$, where *not flying* is the exception solution, for example, only two patterns, $pattern\ (Penguin, motility, not\ flying)$ and $pattern\ (Bird,\ motility,\ flying)$, stored in ASM include the input pattern. These two patterns therefore can be regarded, respectively, as $pattern(c, p, v)$ and $pattern\ (c_0, p, v_0)$ in (6), where $c = Penguin$, $c_0 = Bird$, $p = motility$, $v = not\ flying$, and $v_0 = flying$, By *Lemma 1*, $pattern(c, p, v)$ is recalled first. That is, $P_{OPT1} = pattern(Penguin, motility, not\ flying)$. Because $P_{OPT1} \supseteqq P_{IPT}$, which means that the input pattern has been recovered perfectly, it is therefore not needed to proceed to the second stage. Thus, v (= *not flying*) can be obtained as the solution.

4.4 Recognition inference

In this subsection we give a general analysis case to show that ASM can correctly perform recognition inference via pattern segmentation and recovery.

For a solvable recognition inference problem, $(?, p_1, v_1)$ & $(?, p_2, v_2)$ & $\cdots$ & $(?, p_m, v_m)$ ($m \geqq 1$), we assume c to be the correct solution that has all the given (p_i, v_i) for $i = 1, \cdots, m$. At the first stage, the input to AAM is $P_{IPT} = P_I = pattern\ (\#, \sum_{i=1}^m p_i, \sum_{i=1}^m v_i)$ and both pattern recovery and pattern segmentation are possible. In general the patterns for triples stored in ASM that respectively include $pattern(p_i, v_i)$ ($i = 1, \cdots, m$) can be written as follows:

$$pattern(\{c, S_i\}, p_i, v_i),\ i = 1, \cdots, m, \tag{8}$$

where $\{c, S_i\}$ is the set of concepts that have (p_i, v_i). S_i is the subset excluding c, and $pattern(\{c, S_i\}) = pattern(c) + pattern(S_i)$. All patterns included in (8) compete with each other when the problem is given, and one of them will be recalled via pattern segmentation and pattern recovery. Without loss of generality, we can assume this pattern is $pattern(\{c, S_1\}, p_1, v_1)$ ($= P_{OPT1}$). Because $P_{OPT1} \supseteqq P_{IPT}$ does not hold, ASM proceeds to the second stage, where the input to AAM is $P_{IPT} = P_I + P_{OPT1} = pattern(\{c, S_1\},$ $\sum_{i=1}^m p_i,\ \sum_{i=1}^m v_i)$ and only pattern segmentation is possible, i.e., only a subpattern of P_{IPT} can be recalled.

Lemma 2: For $P_{IPT} = pattern(\{c, S_1\},\ \sum_{i=1}^m p_i,$ $\sum_{i=1}^m v_i)$ at the second stage (i.e., only pattern segmentation is possible and $(\{c, S_1\}, p_1, v_1)$ was recalled at the first stage), a pattern that has the single concept c, i.e., $pattern(c, p_x, v_x)$ ($x \neq 1$) is recalled first.

Proof: Appendix B.

The correct solution c can thus be obtained.

Let us assume the conceptual hierarchy shown in Fig. 1 is stored in ASM. For the recognition problem *(?, motility, running)* & *(?, speed, fast)*, for example, only two patterns, $pattern(\{Mammal, Elephant, Tiger\},$ $motility,\ running)$ and $pattern(\{Tiger, Swallow\},$ $speed,\ fast)$, in ASM include the pattern for each clue, respectively. By matching the two patterns with those shown in (8), they can, respectively, be rewritten as $pattern(\{c, S_1\}, p_1, v_1)$ and $pattern\ (\{c, S_2\}, p_2, v_2)$, where $c = Tiger$, $S_1 = \{Mammal, Elephant\}$, $S_2 = Swallow$, $(p_1, v_1) = (motility, running)$, and $(p_2, v_2) = (speed, fast)$. Thus, c ($= Tiger$) can be obtained as the solution.

4.5 Unknown, clashing, and ambiguous problems

Besides the solvable problems that were described in Sec. 4.3 and 4.4, there are also problems that cannot be solved using existing knowledge (*unknown problems*), problems in which the given clues are insufficient to provide unique answers (*ambiguous problems*), and problems in which the given clues conflict with each other (*clashing problems*). In this subsection we show how ASM rejects unknown and clashing problems by outputting a null pattern and gives ambiguous answers to ambiguous problems by outputting a pattern consisting of several solutions.

For unknown inheritance problem $(c, p, ?)$, i.e., $P_I = pattern(c, p, \#)$, there are no patterns for triples that include P_I and are stored in ASM. But there are patterns for triples that only include c or p instead, provided that all individual concepts, properties, and values given by such a problem are known ones, i.e., they appear in the stored conceptual hierarchy. These patterns compete with each other and one of them is recalled. Note that if a problem includes unknown concepts, properties, or values, it is rejected in the encoding stage. Without loss of generality, we can assume the pattern, P_{OPT1}, recalled at the first stage is:

$$pattern(c_{oth}, p, v_{oth1}) \text{ or } pattern(c, p_{oth}, v_{oth2}), \quad (9)$$

where $pattern(c_{oth}) \supseteqq patern(c)$ does not hold, $p \neq p_{oth}$, and v_{oth1} and v_{oth2} are, respectively, the values of p and p_{oth}. Because $P_{OPT1} \supseteqq P_I$ does not hold, ASM proceeds to the second stage where $P_{IPT} = P_I + P_{OPT1} = pattern(c+c_{oth}, p, v_{oth1})$ or $pattern(c, p+p_{oth}, v_{oth2})$ and only pattern segmentation is possible, i.e., only a subpattern of P_{IPT} can be recalled.

Lemma 3: If $P_{IPT} = pattern(c + c_{oth}, p, v_{oth1})$ or $P_{IPT} = pattern(c, p + p_{oth}, v_{oth2})$ at the second stage, $pattern(c, \#, \#)$ or $pattern(\#, p, \#)$ is recalled.

Proof: Appendix C.

The obtained pattern for solution (value) is therefore null.

For unknown recognition problem $(?, p_1, v_1)$ & $(?, p_2, v_2)$ & $\cdots$ & $(?, p_m, v_m)$ $(m \geqq 1)$, i.e., $P_I = pattern(\#, \sum_{i=1}^{m} p_i, \sum_{i=1}^{m} v_i)$, there are no concepts having all the given (p_i, v_i) $(i = 1, \cdots, m)$ in the stored conceptual hierarchy. The same as the analysis in Sec. 4.4, at the first stage, a pattern, P_{OPT1}, including one of the given (p_i, v_i) $(i = 1, \cdots, m)$ will be recalled. Without loss of generality, we can assume $P_{OPT1} = pattern(c_1, p_1, v_1)$, where c_1 may be a single concept or a set of concepts. Thus, at the second stage, the input is $P_I + P_{OPT1} = pattern\ (c_1, \sum_{i=1}^{m} p_i, \sum_{i=1}^{m} v_i)$. Because the problem is unknown, without loss of generality we can assume that (p_i, v_i) for $i = 1, \cdots, l$ $(1 \leqq l < m)$ are held by c_1, whereas (p_i, v_i) for $i = l+1, \cdots, m$ are not held by c_1. This means the following learned patterns compete with each other:

$$pattern(c_1, p_i, v_i), i = 1, \cdots, l, \quad (10)$$

and

$$pattern(\#, p_i, v_i), i = l+1, \cdots, m. \quad (11)$$

Because the patterns of (11) receive inhibitions from $pattern(c_1)$, whereas those of (10) do not, one of the patterns of (11) will be recalled (segmented) by *Corollary 4*. Consequently, the obtained pattern for the solution (concept) is null.

For clashing problem $(?, p_1, v_1)$ & $(?, p_2, v_2)$ & $\cdots$ & $(?, p_m, v_m)$ $(m \geqq 1)$, where at least two properties p_i and p_j $(i \neq j)$, exist such that $p_i = p_j$ but $v_i \neq v_j$, there is clearly no concept that can hold all the given (p_i, v_i) $(i = 1, \cdots, m)$. These kinds of problems can therefore be regarded as a special case of the unknown recognition problem. Thus, the obtained pattern for a solution (concept) is also null.

For ambiguous problem $(?, p_1, v_1)$ & $(?, p_2, v_2)$ & $\cdots$ & $(?, p_m, v_m)$ $(m \geqq 1)$, there is more than one concept $c_1, \cdots, c_l$ $(l \geqq 2)$ having all the given (p_i, v_i) $(i = 1, \cdots, m)$. If we introduce denotation c to express the set $\{c_1, \cdots, c_l\}$, then the same as the analysis in Sec. 4.4, $pattern(c)$ will be obtained as a final solution (concept). Because the c consists of more than one concept, $c_1, \cdots, c_l$ $(l \geqq 2)$, the answer is ambiguous.

4.6 Computation time for inference

According to Ref. [14], the computation time for pattern segmentation and recovery of AAM tends to be sub-linearly proportional to the number of patterns in competition. Because ASM makes inference by pattern segmentation and recovery, the computation time for inference therefore tends to be sub-linearly proportional to the number of patterns in competition. For most inheritance inference problems, because the patterns shown in (7) do not exist, and only at most two patterns are in competition as shown in (5) and (6), the computation time does not vary with the size of the conceptual hierarchies. Similarly, for the recognition inference problem, the number of patterns in competition only depends on the input complexity (i.e., the number of given clues, or, the number of given pairs of properties and values), as shown in (8). Therefore, the computation time for recognition problems does not vary with the size of the conceptual hierarchy and is sub-linearly proportional to the input complexity. Even for those rare inheritance inference problems in which the patterns shown in (7) do exist, because $m + 2 \leqq D$, where D is the depth of (i.e., the number of levels in) the conceptual hierarchy, the computation time for such problems is at most sub-linearly proportional to the depth of the conceptual hierarchy.

4.7 Flexibility

In ASM, modifying the knowledge is very easy. Adding a new triple, such as *(Elephant, body, large)*, is done using one-shot relearning according to $(L-1)$ without the need to relearn the entire conceptual hierarchy. Deleting triple (c, p, v) is also done using one-shot relearning by restoring the inhibitory connections between $pattern(c)$ and $pattern(v)$. The relearning algorithm for deletion can be written as follows:

> When the pattern of the triple to be deleted is presented, for any units i and j which are in parts CON and VAL, respectively,

$$w_{ji} = \begin{cases} W & \text{if } P_i \cdot P_j > 0 \\ w_{ji} & \text{if } P_i \cdot P_j = 0, \end{cases} \qquad (L-2)$$

where W is an initial value that is the same as that of $(L-1)$. Note that to delete triple (c,p,v), the inhibitory connections between CON and PROP (i.e., between $pattern(c)$ and $pattern(p)$) and between PROP and VAL (i.e., between $pattern(p)$ and $pattern(v)$) cannot be restored. Otherwise, the patterns of other triples that include (c,p) or (p,v) and that are also stored in ASM will be destroyed. Note that the learning algorithm $(L-1)$ and the deleting algorithm $(L-2)$ are not symmetry. Intuitively, the deleting algorithm $(L-2)$ can be comprehended as one that cuts the link between concept and value in the loop-train of concept, property, and value so that the triple cannot be recalled no more as a whole.

ASM also has a generalizing ability. Assuming $c_1, \cdots, c_n$ to be the concepts rooted to c', we have $pattern(c') = \sum_{i=1}^{n} pattern(c_i)$. If new property (p_{new}, v_{new}) is added to $c_1, \cdots, c_n$, by $(L-1)$ the inhibitory connections between (p_{new}, v_{new}) and $pattern(c_i)$ for $i = 1, \cdots, n$ are removed, which means that those between (p_{new}, v_{new}) and $pattern(c')$ are also removed. As a result, ancestor c' will acquire the new property (p_{new}, v_{new}) automatically. If a new property, *(nutritive source, organic matter)*, is added to *Mammal*, *Fish*, and *Bird* in Fig. 1, for example, then their ancestor, *Animal*, will automatically obtain this new property.

Needless to say, there are also demerits in ASM caused by the one-shot learning algorithm under trade off with its merits as described above. Because the weights learned by this algorithm are binary, i.e., they take 0 or a fixed positive value, for example, ASM cannot deal with probabilistic representations which are also needed in handling real-world problems.

5. Computer Simulations

To confirm whether ASM behaves as expected in solving inference problems and to evaluate how fast it performs inference, we conducted two kinds of computer experiments. The parameters we used were as follows. Adaptive weight α was set to 1.2, and bias b (at the first stage) was set to 1.0. Connecting coefficient c was set to 19, to which a random value from 0 to 0.00001 was added. The random value was set as small as possible so that it would not bias the competitive power of each pattern in competition. Initial weight W was set to 20, and the gain of RCU, W_{RI}, was set to the same value as W. To prevent the units of AAM from being too large before AAM reached a stable state, units s_i were assumed to have a saturation value of 10. All units s_i were normalized by this value at each step.

5.1 Behavior of ASM

This experiment was conducted to confirm whether ASM behaves as expected, using the conceptual hierarchy shown in Fig. 1 as an example. The results showed that for all solvable inheritance and recognition problems, no matter whether there are exceptions or not, correct answers can always be obtained. For the problem $(Swallow, motility, ?)$, the answer $flying$ and for the problem $(Penguin, motility, ?)$, the answer *not flying*, for example, were obtained. Also, for the problem *(?, respiration, pulmonary respiration)* & $(?, motility, running)$ & $(?, speed, fast)$, for example, the answer $Tiger$ were obtained. On the other hand, for unknown or clashing problems, the obtained solutions are always null. When an unknown problem, e.g., $(Penguin, speed, ?)$, or a clashing problem, e.g., *(?, respiration, pulmonary respiration)* & *(?, respiration, branchial respiration)*, was given, the obtained pattern for solution (value or concept) was null and the symbolic answer was thus null. For ambiguous problems, the answers were always ambiguous. For the ambiguous problem *(?, respiration, pulmonary respiration)* & *(?, speed, fast)*, for example, a composite pattern composed of the patterns for $Tiger$ and for $Swallow$ was obtained, and a symbolic answer that included both $Tiger$ and $Swallow$ was thus obtained. The number of computation steps needed to obtain the answer depended on both the type of problem and its complexity. It was smaller than 20 for all problems we tested. Our experiments also confirmed the ease of modification (i.e., adding and deleting knowledge by using one-shot relearning) and the generalizing ability (i.e., if a new property is added to all concepts rooted in an ancestor, then this ancestor acquires the new property automatically).

5.2 Computation time of ASM

This experiment was conducted to evaluate how quickly the ASM performs inference. It consisted of two subexperiments: *Experiment I* was performed to examine the relationship between the inference time and the size of the conceptual hierarchy and *Experiment II* was performed to examine the relationship between the inference time and the input complexity.

In *Experiment I*, a conceptual hierarchy with five layers in which the number of concepts was variable was used. As an initial structure, the root concept (the concept at the top layer) had two branches and the concepts from the 2nd through the 4th layer had

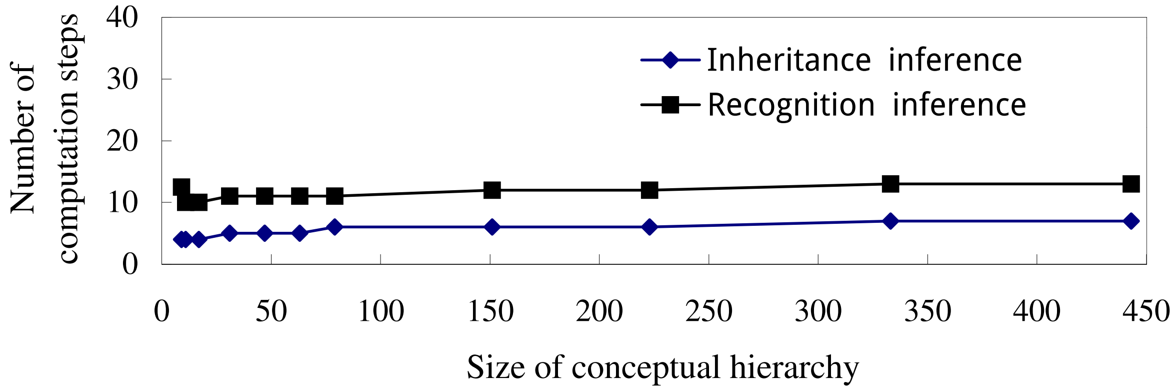

Fig. 5 Relationship between inference time and size of conceptual hierarchy.

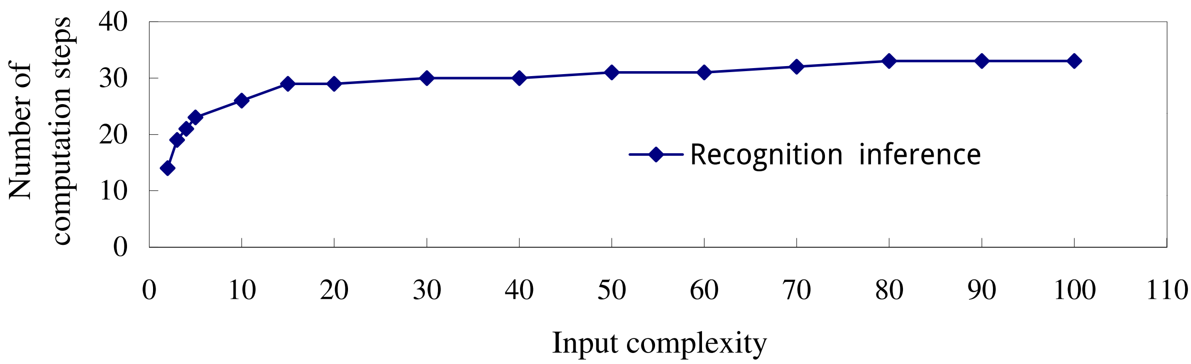

Fig. 6 Relationship between inference time and the input complexity.

one branch each. Each concept had only one pair of property and value. Thus, there were a total of nine concepts and therefore nine triples to be learned in the initial structure. The conceptual hierarchy was then expanded gradually. First, the number of branches for each concept in the 4th, 3rd, and 2nd layer was increased to two in order. Thus, the total number of concepts was increased to 11, 17, and 31 from 9 in order. The number of branches for each concept in the 4th layer was increased from two to eight in increments of two, the number of branches for each concept in the 3rd layer was increased from two to six in increments of two, and the number of branches for each concept in the 2nd layer was increased to four in increments of one in order. Thus, the total number of concepts was further increased to 47, 63, 79, 151, 223, 333, and 443 from 31 in order. For each of these cases, five problems $(c, p, ?)$, where c was a randomly selected concept from the 5th layer and p was a property located in the root concept, were solved to obtain an average computation time for inheritance inference. Also, for each of these cases, five problems $(?, p_1, v_1)\&(?, p_2, v_2)$, where (p_1, v_1) was randomly selected from those in the concepts located in the 5th layer and (p_2, v_2) was randomly selected from those in the concepts located in the 4th layer, were solved to obtain an average computation time for recognition inference. As shown in Fig. 5, the time required to compute both inheritance and recognition inference was nearly unaffected by the size of the conceptual hierarchy. Because the recognition inference requires two processing stages, it took longer than inheritance inference.

In *Experiment II*, a conceptual hierarchy that had the same structure as that used in *Experiment I* and with the maximum of 443 concepts was used. However, the number and location of the properties and their values were different. In this experiment, the properties and values were located only on concepts selected as follows: 1) for each layer only one concept was selected and 2) all the selected concepts were in one train, i.e., were in a hierarchical relationship. For each of the selected concepts, 20 different properties and values were located. Recognition problems that had m pairs of properties and values were solved; m was initially set to 1, then increased until it reached 100 in increments of 1 when $m \leqq 5$, in increments of 5 when $5 < m \leqq 20$, and in increments of 10 when $m > 20$ in order. The properties and values were selected from the stored conceptual hierarchy as follows: those located in the root concept were used first when $m \leqq 20$, and then those located in the concepts of the 2nd, 3rd, and 4th layers were used in order with increasing m. As shown in Fig. 6, the computation time for inference tended to be sub-linearly proportional to the input complexity.

6. Conclusion

We have demonstrated that the adaptive associative memory can be used to construct a new model of semantic networks. The main feature of the proposed model is its computational effectiveness which is an important issue in knowledge representation; it can do inference based on large conceptual hierarchies extremely fast—in time that does not increase with the size of conceptual hierarchies. This performance cannot be realized by any existing models. By this work, it has been verified that not only procedural knowledge, which has been treated in the previous studies, but also semantic knowledge can be well represented within the same framework of the associative memories that can do one-to-many association or pattern segmentation. Such associative memories can therefore be regarded as an essential module of cognitive systems. By memorizing domain specific semantic and procedural knowledge in such associative memories, it can be expected that a computationally effective model to address various cognitive processes such as problem-solving and language understanding can be constructed.

References

[1] D. G. Bobrow and T. Winogard, "An overview of KRL: a knowledge representation language," *Cognitive Science*, 1(1), pp. 3-46, 1977.
[2] R. J. Brachman and J. Schmolze, "An overview of KL-ONE knowledge representation system," *Cognitive Science*, 9(2), pp. 171-216, 1985.
[3] R. L. Chapman, "Roget's international thesaurus, Fifth Edition, HarperPernnial," A division of HarperCollins Publishers, 1992
[4] M. Derthik, "A connectionist architecture for representation and reasoning about structured knowledge," *Proc. Cognitive Science Conference,* 1987.
[5] S. E. Fahlman, "Representing implicit knowledge," In G. E. Hinton & J. A. Anderson (Eds), *Parallel models of associative memory,* Lawrence Erlbaum, 1989.
[6] J. Geller, "Advanced update operations in massively parallel knowledge representation," In H. Kitano & J.A. Hendler (Eds), *Massively parallel artificial intelligence,* Cambridge, MA, AAAI Press/MIP Press, 1994.
[7] M. H. Hassoun, "Associative Neural Memories–Theory and Implementation," Oxford University Press, 1993.
[8] Y. Hirai, "Representation of procedural knowledge by a model of human associative processor, HASP," *Trans. IECE of Japan*, J69-D, pp. 1743-1753, 1986 (in Japanese).
[9] Y. Hirai, "Representation of control structures by a model of associative processor, HASP," *Proc. 1st Annual International Conference on Neural Networks*, pp. II-447-454, 1987.
[10] Y. Hirai and Q. Ma, "Modeling the process of problem-solving by associative networks capable of improving the performance," *Biol. Cybern.*, vol. 59, pp. 353-365, 1988.
[11] G. E. Hinton, "Implementing semantic networks in parallel hardware," In G. E. Hinton & J. A. Anderson (Eds), *Parallel models of associative memory,* Lawrence Erlbaum, 1989.
[12] J. J. Katz and J. A. Fordor, "The structure of a semantic theory," *Language*, 39, pp. 170-210, 1963.
[13] Q. Ma and Y. Hirai, "Modeling the acquisition of counting with an associative network," *Biol. Cybern.*, vol. 61, pp. 271-278, 1989.
[14] Q. Ma, "An adaptive type of competitive network for pattern segmentation in associative memory," *IEICE of Japan*, Tech. Rep. NC93-64, pp. 75-82, 1993 (in Japanese).
[15] Q. Ma, "Connectionist realization of semantic networks using an adaptive associative memory, AAM," *1995 IEEE International Conference on Neural Networks Proc. (ICNN'95),* pp. 1328-1333.
[16] Q. Ma, "Adaptive associative memories capable of pattern segmentation," *IEEE Trans. Neural Networks,* vol. 7, no. 6, pp. 1439-1449, 1996.
[17] D. Marr, "Vision," Freeman, 1982.
[18] M. R. Quillian, "Semantic memory," In M. Minsky (Ed.), *Semantic information processing,* Cambridge, MA: MIT Press, 1968.
[19] P. G. Schyns, "A modular neural network model of concept acquisition," *Cognitive Science*, 15, pp. 461-508, 1991.
[20] L. Shastri, "A connectionist approach to knowledge representation and limited inference," *Cognitive Science,* 12, pp. 331-392, 1988.
[21] L. Shastri, "Why semantic networks?," In J. F. Sowa (Ed), *Principles of semantic networks,* Morgan Kaufmann Publishers, 1991.
[22] E. E. Smith and D. L. Medin, "Categories and concepts," Cambridge, MA: Harvard University Press, 1981.
[23] M. Zeidenberg, "Neural networks in artificial intelligence," Ellis Horwood, 1990.

Appendix A: Appendix

Proof of Lemma 1:

(a) Because all patterns in (5)-(7) have the same property p and are stored ones, there are no inhibitory connections between the 1-bits in $pattern(p)$ and the 1-bits in $pattern(x)$ where $x = c, c_i, v, v_i$ for $i = 0, \cdots, m$ by $(L-1)$. Thus, $pattern(p)$ does not affect the competition.

(b) We can show that the 1-bits in $pattern(v)$ receive more inhibitions than the 1-bits in $pattern(v_i)$ for $i = 0, \cdots, m$ as follows.

When the patterns in (5) and (7) compete with each other, because all of them are stored patterns, the 1-bits in $pattern(v_i)$ $(i = 1, \cdots, m)$ do not receive inhibitions from the 1-bits in $pattern(c_i)$ $(i = 1, \cdots, m)$ by $(L-1)$. Further, because $pattern(c_i) \supseteqq pattern(c_0) \supseteqq pattern(c)$ $(i = 1, \cdots, m)$, the 1-bits in $pattern(v_i)$ $(i = 1, \cdots, m)$ also do not receive inhibitions from the 1-bits in $pattern(c_0)$ and $pattern(c)$. Therefore, the 1-bits in $pattern(v_i)$ $(i = 1, \cdots, m)$ only receive inhibitions from the 1-bits in $pattern(c_j)$ for $j = 1, \cdots, m$ and $j \neq i$. On the other hand, the 1-bits in $pattern(v)$ receive at least inhibitions from the 1-bits in $pattern(c_i)$ for $i = 1, \cdots, m$. Therefore, the 1-bits in $pattern(v)$ receive more inhibitions than the 1-bits in $pattern(v_i)$ for $i = 1, \cdots, m$.

Similarly, when the patterns in (6) and (7) compete with each other, the 1-bits in $pattern(v)$ receive inhibitions from $pattern(c_i)$ for $i = 0, \cdots, m$, but the 1-bits in $pattern(v_i)$ $(i = 0, \cdots, m)$ only receive inhibitions from $pattern(c_j)$ for $j = 0, \cdots, m$ and $j \neq i$. Therefore, the 1-bits in $pattern(v)$ also receive more inhibitions than the 1-bits in $pattern(v_i)$ for $i = 0, \cdots, m$.

By (a), (b), and *Corollary 4*, *Lemma 1* is proved. □

Appendix B: Appendix

Proof of Lemma 2:

(a) If S_1 in P_{IPT} is null, $pattern(c, p_i, v_i)$ for $i = 2, \cdots, m$ compete with each other and $pattern(c, p_x, v_x)$ $(2 \leqq x \leqq m)$ is recalled. Note that (c, p_1, v_1), which was recalled at the first stage, cannot be recalled again.

(b) If S_1 in P_{IPT} is not null, because the answer exists uniquely, among $S_2, \cdots, S_m$ in (8) there is at least one S-set in which all concepts are not those of S_1. Without loss of generality, we can assume S_m is such a set. Thus, the inhibitory connections between the 1-bits in $pattern(p_m, v_m)$ and the 1-bits in the patterns for the concepts of S_1 are intact after learning by $(L-1)$. The following $m-1$ stored patterns therefore compete with each other:

$$pattern(c, p_m, v_m), \tag{B-1}$$

$$pattern(\{c, S_i^{(1)}\}, p_i, v_i), \ i = 2, \cdots, m-1. \tag{B-2}$$

Here $S_i^{(1)}$ is a subset of S_1 and its element has (p_i, v_i). Note that $S_i^{(1)}$ may be null. Among these patterns, the 1-bits in $pattern(p_m, v_m)$ [in $(B-1)$] receive inhibitions from all patterns for the concepts in S_1 as noted above, whereas the 1-bits in $pattern(p_i, v_i)$ $(i = 2, \cdots, m-1)$ [in $(B-2)$] at least do not receive inhibitions from the concepts in $S_i^{(1)}$. Thus, if no $S_i^{(1)}$ $(i = 2, \cdots, m-1)$ is null, the 1-bits in $pattern(p_m, v_m)$ receive more inhibitions, and $pattern(c, p_m, v_m)$ is recalled by *Corollary 4*. If there are some $S-sets$, $S_l^{(1)}$ $(2 \leqq l \leqq m-1)$, that are null, either the 1-bits in $pattern(p_m, v_m)$ or the 1-bits in $pattern(p_l, v_l)$ $(2 \leqq l \leqq m-1)$ receive more inhibitions, and therefore either $pattern(c, p_m, v_m)$ or $pattern(c, p_l, v_l)$ $(2 \leqq l \leqq m-1)$, i.e., $pattern(c, p_x, v_x)$ $(x \neq 1)$, is recalled first by *Corollary 4*. *Lemma 2* is thus proved. □

Appendix C: Appendix

Proof of Lemma 3:

If $P_{IPT} = pattern(c + c_{oth}, p, v_{oth1})$ at the second stage, (a) neither $pattern\ (c_{oth}, p, v_{oth1})$, which was recalled at the first stage, nor all its subpatterns can be recalled again; (b) $pattern(c, p, v_{oth1})$ cannot be recalled, because it is not stored in ASM; (c) $pattern\ (c, p, \#)$ can not be recalled, because there are inhibitory connections between $pattern(c)$ and $pattern(p)$. Conversely supposing there are no inhibitory connections between $pattern(c)$ and $pattern(p)$, this means that there is a pattern $pattern(c\ or\ c_0, p, v_x)$ stored in the ASM, where c_0 is ancestor of c. This is contradictory to the definition of unknown problems; (d) $pattern(c, \#, v_{oth1})$ can not be recalled, because there are inhibitory connections between $pattern(c)$ and $pattern(v_{oth1})$. Conversely supposing there are no inhibitory connections between $pattern(c)$ and $pattern(v_{oth1})$, this means there is a pattern $pattern(c\ or\ c_0, p_x, v_{oth1})$ stored in the ASM, where c_0 is ancestor of c and $p_x \neq p$. Because this pattern cannot be the same one of the patterns shown in (9), the previous supposition is false. By (a)-(d), the uniquely possible pattern that can be recalled (segmented) is $pattern(c, \#, \#)$. Similarly, if $P_{IPT} = pattern(c, p + p_{oth}, v_{oth2})$ at the second stage, the uniquely possible pattern that can be recalled (segmented) is $pattern(\#, p, \#)$.

Lemma 3 is thus proved. □

Qing Ma received B.S. degree in electrical engineering from Beijing University of Aeronautics and Astronautics, China, in 1983, and the M.S. and Dr. Eng. degree in computer science from University of Tsukuba, Japan, in 1987 and 1990. Since 1993 he has been working at the Communications Research Laboratory, Ministry of Posts and Telecommunications, Japan, currently as a Senior Researcher. His research interests include neural networks and its applications in knowledge representation and natural language processing.

Hitoshi Isahara received B.E., M.E. degrees in electrical engineering from Kyoto University, Japan, in 1978 and 1980 respectively, and Dr. Eng. degree in electrical engineering from Kyoto University, Japan, in 1995. He worked in Electrotecnical Laboratory, Ministry of International Trade and Industry, Japan, from 1980 to 1995. In 1995, he moved to the Communications Research Laboratory, Ministry of Posts and Telecommunications, Japan, where he is now the Chief of the Intelligent Processing Section. His research interests include natural language processing, AI, and knowledge representation.