Modeling the Cognitive Processes with Associative Networks

Qing Ma

Institute of Information Sciences and Electronics

Submitted to the University of Tsukuba, Japan
in partial fulfillment of the requirements
for the degree of
Doctor of Engineering

1989

Contents

Chapter 1

Introduction

1.1 Background of Studies of Neural Networks

Although modern high-speed computers operate with processing units that function on the order of tens or hundreds of nanoseconds, the brain, which consists of processing units that operate on the order of milliseconds, is far quicker and better at perceiving objects in natural scenes and finding their relations, at accessing whatever information we need from memory and understanding langauge, at making plans and carring out appropriate actions, and at wide range of other natural cognitive tasks. We human being are also far better at learning to do these things more fluently and effectively through practice.

The basis for these differences between today's computers and the brain may partially come from "software", the traditional one we might expect from artificial intelligence. However, in our view, people are smater than computers mainly because that the brain employs a very different computational architecture that is more suited to deal with the natural information processing tasks that people are so good at.

In order to understand the complex information processing systems such as the brain, we must understand them at three levels (Marr, 1982). The top level is the abstract computational theory of the devices, in which the performance of the device is characterized as a mapping from one kind of information to another, the abstract properties of this mapping are defined precisely, and its appropriateness and adequacy for the task at hand are demonstrated. The central level is to understand how this computational theory can be implemented, in particular, how the representation for the input and output and the algorithm to be used to transform one into the other are chosen. The last level is the details of how the algorithm and representation are implemented in hardware.

According to Marr's theory described above, for understanding the basic computational architecture of the brain and the mystery of human cognitive abilities it seems very essential to study massively parallel and highly interconnected networks of neuron-like elements, which are inspired by the obvious features, such as parallelism and distribution, of the brain (Anderson & Hinton, 1981). The networks are variably referred to as neural networks, connectionist networks, parallel distributed processing systems, and neural computers. These studies attempt to understand the mechanism of the brain by modeling its parts and their connections which can perform verious kinds of cognitive tasks or can explain various meaningful phenomena observed in human cognitive processes, and trying to characterize mathematically the kinds of useful computation that such assemblies perform.

The history of studies of neural networks can be traced back to *perceptron* proposed by Rosenblatt (1962) and the neural networks have successfully been em-

ployed to deal with a variety of problems in vision, word perception, associative memory, word sense disambiguation, speech production, learning and so on. Especially in recent years, a very powerful learning algorithm called *backpropagation* has been proposed (Rumelhart, Hinton, & Williams, 1986) and has been sucessfully applied in the wide range of cognitive tasks. Moreover, it has been demonstrated that the calculations such as the decisions in combinational optimization problems, which take tremendously long time for current computers, can be accomplished by neural networks very rapidly and effectively (Hopfield & Tank, 1985). Because the power of neural networks has been developed successfully and recognized gradually, the expectation and interest in the studies of neural networks is rapidly growing.

1.2 Memory and Cognitive Processes

Without memory, there can be no perception as we experience it, no learning, no motivated action – Almost all of the cognitive activities depend on the contextually and effetively retrieval of the necessary information from memory. Therefore, to understand the architecture of human memory and how the memory has to do with the various kinds of cognitive functions are very important goals of the studies of the brain. At the same time, it can provide a very important clue for the understanding of the whole mechanism of the brain.

In psychology, human memory is divided into three processes, which are recording, storage and retrieval, and is classified into two kinds, the short term memory and long term memory (Waugh & Norman, 1975; Norman, 1969). Moreover, according to Tulving's theory (1983), the long term memory is classified into the

procedural memory and the propositional memory, the latter of which is further classified into the episodic memory and the semantic memory. The episodic memory is defined as the memory for specific, personally experienced events, while the semantic memory is the memroy for general principles, associations, rules, and the like. The procedural memory, which is also called operational memory, consists of the skills of perceptual motion or cognition. In other words, it is the memory of procedural steps for performing cognitive tasks such as the operation of machines or the driving of cars.

So far, mainly two kinds of models for the semantic memory have been proposed. One is AI models that typified by the semantic network (Quillian, 1969). In these models, the semantic knowledge is represented by the nodes that express concepts and the links that express the relations between the concepts. The other is associative networks which are modeled by considering the fact that the brain is constructed by hardware called neural network and human memory is characterized by associative processing (Kohonen, 1978). As the models of the procedural memory, the procedural network (Brown & Burton, 1978) and the production system (Greeno, 1978; Anderson, 1983) are well known. Recently, the associative network model has been developed to represent the procedural memroy (Hirai 1986; Hirai & Ma, 1988).

The idea that a prominent feature of human memory is the association can be traced back as far as Aristotle (Anderson & Bower, 1973). However, only in the last twenty years has the associative memory idea been adopted in the modeling of human memory and various kinds of models of associative memory have

been proposed. Among them, the most noticeable and widely employed one is the correlation matrix memories proposed by Kohonen (1972). His correlation matrix formalism is described in the next section, because it is also the mathematical basis of our models to be proposed in this dissertation.

1.3 Correlation Matrix Memories and Neural Networks

Let us consider l pairs of key vectors and associated vectors and let the kth pair of them be $X^{(k)} = (x_1^{(k)}, x_2^{(k)}, \ldots, x_m^{(k)})$ and $Y^{(k)} = (y_1^{(k)}, y_2^{(k)}, \ldots, y_n^{(k)})$, respectively. The correlation matrix M that memorizes the l pairs of associative relationships is defined by

$$M = \sum_{k=1}^{l} X^{(k)T} \cdot Y^{(k)}, \tag{1.1}$$

where T designates transposition of a vector. The process of associative recollection is described follows:

$$Y = X \cdot M = X \cdot \sum_{k=1}^{l} X^{(k)T} \cdot Y^{(k)}, \tag{1.2}$$

where X is a key input and Y is a recollected vector. If $X = X^{(k)}$, we have

$$Y = X^{(k)} \cdot X^{(k)T} \cdot Y^{(k)} + \sum_{i \neq k} X^{(k)} \cdot X^{(i)T} \cdot Y^{(i)}. \tag{1.3}$$

If the key vectors are mutually orthogonal, the inner products $X^{(k)} \cdot X^{(i)T}$ disappear and the associated vector $Y^{(k)}$ appears in proportion to the inner product $X^{(k)} \cdot X^{(k)T}$. If the key vectors are not orthogonal, crosstalk noise $Y^{(i)}$ appers in proportion to $X^{(k)} \cdot X^{(i)T}$.

In general, a neural network consists of a large number of neuron-like simple computing elements connected via synapse-like weighted links. The elements are

computational entities defined by a real-value potential. A element communicates with the rest of the network by transmitting a single output value to all elements it is connected to. The elements receive inputs via weighted links. Each link contributes an input whose magnitude equals the output of the each element at the source of the link times the weight on the link.

The correlation matrix memory described above can be modeled by a basic neural network, in which the transmiting efficiency (weighted links) from key input elements to output elements corresponds to the magnitude of elements of the correlation matrix (e.g. Fukushima, 1979, 1989). In fact, by introducing inhibitory element to the basic neural network or seting up threshold on the elements, the crosstalk noise described above can be taken away or can be reduced.

However, crosstalk noise will necessarily occur when one and the same key is associated with more than one item. In this case, (1.3) becomes

$$Y = X^{(k)} \cdot X^{(k)T} \cdot (Y^{(k1)} + Y^{(k2)} + \cdots + Y^{(kh)}), \tag{1.4}$$

provided that the key vectors are mutually orthogonal. In order to resolve the multiple-match problem, a model of associative network, HASP, has been proposed by Hirai (1983).

1.4 Purposes and Outline of the Dissertation

The purposes of this dissertation are to represent three kinds of cognitive processes, including problem-solving, knowledge representation, and learning, with neural networks and to demonstrate that all these neural networks can be constructed withing the framework of an associative network, HASP. The research is based on the belief

that the memory is the most fundamental and essential function in human cognitive processes. In Chapter 2, we model the process of problem solving and show that the performance of the model can be improved by the priming effect and the merging between procedural steps. To our knowledge, so far, no such neural network models have been proposed. Chapter 3 describes a new scheme for knowledge representation. Some of defects existing in the previous models can be improved and the results of computer simulation show that our model can handle knowledge very quickly and effectively. In Chapter 4, we propose a neural network which can mimic some aspects of learning of sequence and provide an explanation for the psychological phenomena observed in children's learning process. Finally, in Chapter 5, we make a brief conclusion for our works and present future directions of the research.

Chapter 2

Modeling the Process of Problem-Solving

In this chapter we propose a model of neural network underlying arithmetic problem-solving. All of the models for procedural memory, semantic memory, and working memory, which are necessary to solve problems and are basic constituents composing the model, are constructed within the framework of an associative network, HASP. Performance of the model is simulated on a digital computer. By memorizing primitive knowledge of addition of two digits such as $6 + 8 = 14$ in the semantic memory and by memorizing procedural knowledge for the control of the process of adding in the procedural memory, the model can perform addition of multiple numbers with multiple digits. The performance of the model can be improved by making explicit serial associative relationships between consecutive procedural steps, because a current procedural step primes the next one. In addition, if a preceding procedural step is the subset of the next one, merging between the two steps occurs. The performance can be improved about 20% by the priming ef-

fect and the merging. By memorizing incorrect procedures, the model can also generate four kinds of bugs that were observed in children's practice of addition.

2.1 Motivation

Our cognitive activities, such as problem solving, are carried out by using various kinds of knowledge stored in various memory systems. When we solve an arithmetic problem such as adding two numbers with multiple digits, for example, we use primitive knowledge of addition of two digits such as $6+8=14$ stored in a semantic memory, procedural knowledge stored in a procedural memory which controls the digit-by-digit and column-by-column solution process of the adding, and working memory that temporarily stores intermediate results to be used later on demand. Therefore, to understand how such problem solving tasks are carried out in our brains, first we must understand individual memory systems which can store and retrieve relevant knowledge, and then we must understand how to combine them to solve problems in cooperation.

Since the neural networks underlying individual memories are composed of a large number of low-speed components, neurons, the high-speed retrieval time of our memories must come from their parallel and distributed structures and the suitable computations they perform. Therefore, to understand cognitive systems it is essential to figure out their computational architectures underlying individual memories and their combinations.

Psychological studies on problem solving were carried out earlier by Gestalt psy-

chologists. Many psychological models on solution processes of problems, ranging from puzzles such as tower of Hanoi to arithmetic and geometric problems, have been proposed by using production systems (Newell & Simon, 1972; Greeno, 1978; Anderson, 1980; Anderson, 1983; Kintsch & Greeno, 1983).

This chapter presents a model of neural network underlying arithmetic problem solving. Since our memories are characterized by associative processing, all of the memory models constituting the present model are constructed within the framework of HASP, a model of human associative processor proposed by Hirai (1983). One of the most prominent features of HASP is its capability of memorizing one-to-many associative relationships. By this function, several items associated with one particular key can be retrieved one by one, and set-theoretically defined multiple key search operations such as intersection, join, difference and negation, as well as partial key search operation can be performed. Especially set intersection operation can be carried out by providing all relevant keys in parallel and only items associated with all those keys simulateneuosly can be retrieved.

The structure of this chapter is as follows. In Section 2.2, we describe HASP with positive feedback, which is used to construct procedural memory. Section 2.3 presents overall structure of the model. In Section 2.4, control structure of the process of adding is described. In Section 2.5 and 2.6, the models of working memory and semantic memory are described respectively. Section 2.7 shows results of computer simulation and presents discussions about improvement in the efficiency of performance. It is also shown that the model can generate four kinds of bugs observed in children's practice.

2.2 HASP with Positive Feedback

In this section we describe HASP with positive feedback, which can store and retrieve procedural knowledge. Retrieval of the procedural knowledge from the associative network can be viewed as the retrieval of an action by a set of conditions in a production rule. Structure of the network is shown in Fig. 2.1. It consists of two components: heteroassociative network with an orthogonalized input, and inhibitory recurrent network. The readout control unit of HASP (Hirai, 1983) is not involved.

2.2.1 Heteroassociative network with an orthogonalized input

In this network, associative relationship between an external key input, which is presented through $K(x)$, and a procedural step, which is presented through $T(y_1)$, is stored in the excitatory modifiable connections $W_{SK}(y_1, x)$. These connections are strengthened from initial strength, which is assumed to be zero, to some positive value, according to the following learning algorithm ($L - 2.1$):

1. if $K(x) \cdot T(y_1) > 0$, $W_{SK}(y_1, x)$ is increased to some positive value W_S, and
2. if $K(x) \cdot T(y_1) = 0$, $W_{SK}(y_1, x)$ is not changed.

The external key input provides context or status information of the model. Besides, the serial associative relationship from a procedural step to the next one is established as follows. Let us assume that the first procedural step is held in HASP by positive feedback loops denoted by $A(y_2)$ in Fig. 2.1. By providing the next

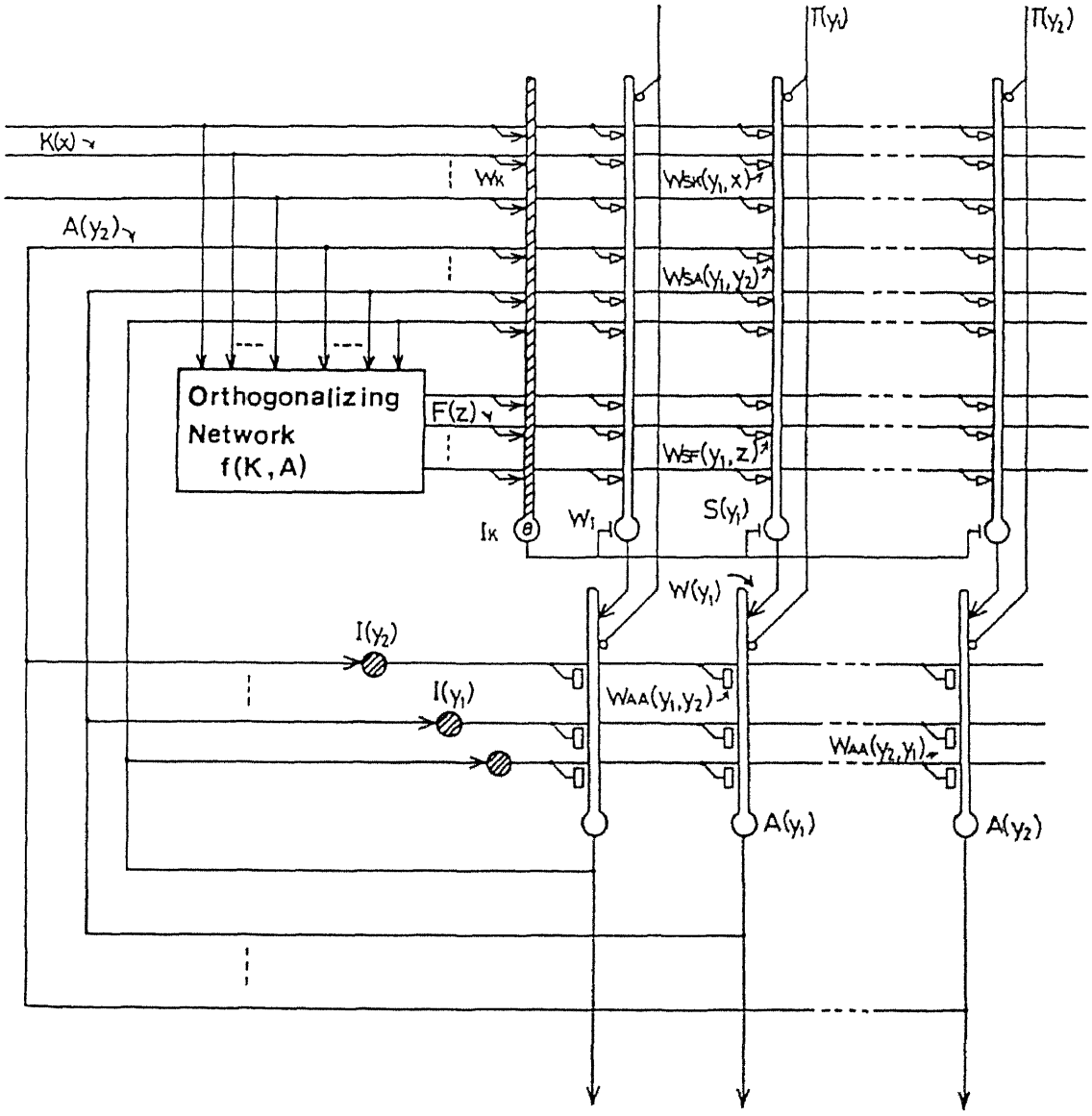


Fig. 2.1. Structure of HASP with positive feedback. Network notations are follows. Open elements: excitatory. Hatch elements: inhibitory. —➔ : fixed excitatory connection. —➔ : modifiable excitatory connection whose strength is increased by learning. —○: excitatory connection which carries teacher signal. —⊣ : fixed inhibitory connection. —▯ : modifiable inhibitory connection whose strength is weakened by learning.

procedural step through $T(y_1)$, the serial associative relationship is stored in the excitatory modifiable connections $W_{SA}(y_1, y_2)$ by strengthening their connections from initial strength to W_S, according to the following algorithm $(L - 2.2)$:

1. if $A(y_2) \cdot T(y_1) > 0$, $W_{SA}(y_1, y_2)$ is increased to W_S, and

2. if $A(y_2) \cdot T(y_1) = 0$, then $W_{SA}(y_1, y_2)$ is not changed.

By this learning algorithm, each procedural step is also associated with itself, so that a positive feedback path can be established.

An orthogonalizing network is introduced to avoid a partial key problem . Let us consider the following production rules stored in an associative network: ("1", "2", "3") ==> "A", and ("1", "2") ==> "B", where the left hand side of the arrow represents a set of conditions and the right hand side represents an action that should be executed when all of the conditions are satisfied. In this case, the conditions of the second rule is included in the conditions of the first one. This causes partial key problem as described below. Each production rule can be represented by a pair of associative relationship between a set of conditions and an action. Conditions of each rule is considered as spatial key patterns, which are presented to the heteroassociative network through $K(x)$ and $A(y_2)$ in parallel, and an action of each rule is considered as a spatial pattern, which is presented through $T(y_1)$. Therefore, the above two production rules can be represented by an associative mapping function $M[*]$ as follows:

$$M[\text{"1"}, \text{"2"}, \text{"3"}] = \{\text{"}A\text{"}\},$$

and

$$M[\text{“1”}, \text{“2”}] = \{\text{“}B\text{”}\}.$$

According to the characteristic of HASP (Hirai, 1983), the above two relationships can be rewritten in the following form:

$$M[\text{“1”}] = \{\text{“}A\text{”}, \text{“}B\text{”}\},$$

$$M[\text{“2”}] = \{\text{“}A\text{”}, \text{“}B\text{”}\},$$

and

$$M[\text{“3”}] = \{\text{“}A\text{”}\}.$$

Since HASP retrieves a set of actions by the intersection of the sets specified by the relevant conditions, it becomes

$$M[\text{“1”}, \text{“2”}] = M[\text{“1”}] \cap M[\text{“2”}]$$

$$= \{\text{“}A\text{”}, \text{“}B\text{”}\} \cap \{\text{“}A\text{”}, \text{“}B\text{”}\} = \{\text{“}A\text{”}, \text{“}B\text{”}\}.$$

Hence, “B” can not always be retrieved by $M[\text{“1”}, \text{“2”}]$. A simple solution to circumvent such partial key problem is to add unique conditions made from the original conditions by some function $f(*)$, and the two production rules are modified as follows:

$$(\text{“1”}, \text{“2”}, \text{“3”}, f(\text{“1”}, \text{“2”}, \text{“3”})) ==> A,$$

$$(\text{“1”}, \text{“2”}, f(\text{“1”}, \text{“2”})) ==> B.$$

where $f(\text{“1”}, \text{“2”}, \text{“3”})$ is made to not be equal to $f(\text{“1”}, \text{“2”})$. Since $M[f(\text{“1”}, \text{“2”})]$ = “B”, it becomes $M[\text{“1”}, \text{“2”}, f(\text{“1”}, \text{“2”})] = M[\text{“1”}] \cap M[\text{“2”}] \cap M[f(\text{“1”}, \text{“2”})] =$

"B". Hence, B can always be retrieved by("1", "2", f("1", "2")). This function $f(*)$ is implemented by an orthogonalizing network which can transform any input patterns to mutually orthogonal ones. In our model, the orthogonalizing network proposed by Hirai (1984) is employed and it is described in Appendix of (Hirai & Ma, 1988). The associative relationship between the output of the orthogonalizing network and a procedural step, which is presented through $T(y_1)$, is stored in the excitatory modifiable connections $W_{SF}(y_1, z)$ by strengthening their connections from initial strength to W_S, according to the following learning algorithm ($L-2.3$):

1. if $F(z) \cdot T(y_1) > 0$, $W_{SF}(y_1, z)$ is increased to W_S, and

2. if $F(z) \cdot T(y_1) = 0$, $W_{SF}(y_1, z)$ is not changed.

The inhibitory element I_K with threshold θ gathers the activity of key input, positive feedback input and output of the orthogonalizing network through excitatory connections W_K and suppresses output elements $S(y_1)$ of the heteroassociative network through inhibitory connections W_I. The response of the inhibitory element I_K with threshold θ at time t is defined by

$$I_K(t) = \varphi[\sum_x W_K \cdot K(x,t) + \sum_{y_2} W_K \cdot A(y_2,t) + \sum_z W_K \cdot F(z,t) - \theta], \qquad (2.1)$$

where $\varphi[*]$ is a nonlinear transfer characteristic of an analog threshold element defined by

$$\varphi[a] = \begin{cases} a, & \text{if } a > 0 \\ 0, & \text{if } a \leq 0. \end{cases} \qquad (2.2)$$

By the suppression from the element I_K, the output of the network is limited to the magnitude between zero and the threshold in spite of the existence of positive

feedback loops. The function of set intersection operation is also implemented by this suppression (Hirai 1983).

The element $S(y_1)$ is assumed to be an analog threshold element with time lag of the first order. The response is defined by

$$\mu_1 \frac{dS^*(y_1,t)}{dt} + S^*(y_1,t) = \sum_x W_{SK}(y_1,x) \cdot K(x,t)$$

$$+ \sum_{y_2} W_{SA}(y_1,y_2) \cdot A(y_2,t) + \sum_z W_{SF}(y_1,z) \cdot F(z,t) - W_I \cdot I_K(t), \qquad (2.3)$$

and

$$S(y_1,t) = \varphi[S^*(y_1,t)]. \qquad (2.4)$$

2.2.2 Inhibitory recurrent network

At the output of the heteroassociative network, not only a current procedural step, but also to-be-retrieved steps which are associated with both the current one and the current external key pattern appear as a superimposed pattern. The inhibitory recurrent network is used to make the current step suppress the next ones, and only the current one appears at the output of HASP.

Structure of the network is similar to that of autoassociative memory models, but the connections denoted $W_{AA}(y_1,y_2)$ are inhibitory, and the strength is reduced from some initial value W_A to zero according to the following learning algorithm $(L-2.4)$:

1. if $T(y_1) \cdot T(y_2) > 0$, $W_{AA}(y_1,y_2)$ is reduced to zero, and

2. if $T(y_1) \cdot T(y_2) = 0$, then $W_{AA}(y_1,y_2)$ is not changed.

With this algorithm the autocorrelation of a spatial pattern representing each procedural step is stored as a spatial distribution pattern of reduced inhibitions. Hence, if each procedural step is a binary pattern composed of 0 and 1 elements, and at least one 1-element of the pattern is not included in the others, then the inhibitions between such 1-elements will remain. By assuming the initial value to be greater than 1.0, the multiple procedural steps, which appear at the output of heteroassociative network simultaneously, compete through the remaining inhibitory connections, and finally only the current procedural step appears at the output of HASP.

The element $A(y_1)$ is assumed to be an analog threshold element with time lag of the first order. The response of $A(y_1)$ at time t is defined by

$$A(y_1, t) = \varphi[A^*(y_1, t)], \tag{2.5}$$

where

$$\mu_2 \frac{dA^*(y_1, t)}{dt} + A^*(y_1, t) = W(y_1) \cdot S(y_1, t) - \sum_{y_2} W_{AA}(y_1, y_2) \cdot I(y_2, t), \tag{2.6}$$

and

$$I(y_2, t) = A(y_2, t), \tag{2.7}$$

where $W(y_1)$ is a connecting coefficient from $S(y_1)$ to $A(y_1)$.

It should be mentioned that the retrieval of the next procedural step is carried out by changing the external key input instead of using the readout control unit used in (Hirai, 1983).

2.3 Overall Structure of the Model

Overall structure of the model that can represent the process of addition is shown in Fig. 2.2. The box denoted by "input system", together with that denoted by "representation of numerical concepts and operators", constitutes the input encoding subsystem. This subsystem reads and recognizes each digit or the operator written on paper as our vision does, and a specific neural element or a numerical concept encoding the digit or the operator in the latter box is activated. In our model, however, this subsystem is assumed to exist and its modeling is left as a future work. The box denoted by "output system", together with that denoted by "representation of motor images" constitutes the output subsystem. It writes out answer on paper. It also writes out intermediate results such as a carry for later use. In our model, however, this subsystem is assumed to exist and the modeling is left as a future work, too.

A working memory, which is requisite for the construction of the model, is not shown as a separate box , but is embedded in the box denoted by "representation of nemerical concepts and operators". It is used to temporarily store digits and the operator read from paper or intermediate results that will be used later on demand in the process of adding. In our work, it is modeled as an associative network temporarily storing associative relationships between a procedure and a numerical concept or an operator activated by that procedure.

The semantic memory of addition stores and retrieves the primitive knowledge of addition of two digits from 0+0=0 to 9+9=18. HASP with input buffers for augend, addend and operator is used to model this memory. Since the answer is

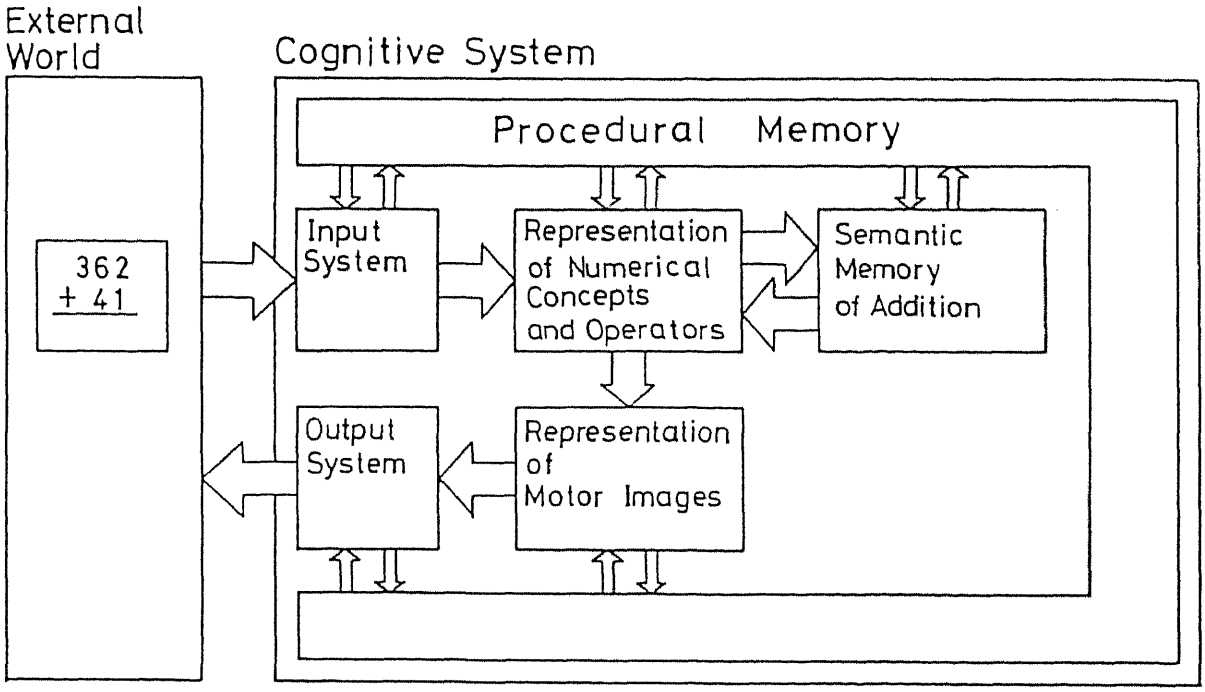


Fig. 2.2. Overall structure of the model.

also a nemerical concept, it is fed back to the box denoted by "representation of numerical concepts and operators", and activates a corresponding network element, so that the intermediate results to be used later can be temporalily stored in the working memory.

The process of adding is controlled by the procedural knowledge stored in the procedural memory system. In the construction of this memory, HASP with positive feedback described in Section 2.2 is employed. According to the overall status of the model, a set of appropriate control signals constituting a micro-procedure is retrieved from the memory and controls the behavior of each part of the model.

2.4 Control Structure of the Process of Adding

Most of the models of problem-solving processes hitherto proposed have been constructed by using production systems(Greeno, 1978). For the process of addition, for example, Anderson used seven production rules to represent the procedural knowledge (Anderson, 1980). Each production rule consists of a set of conditions and a subgoal which should be executed when the set of conditions is satisfied. The retrieval of an executable subgoal is carried out by pattern matching between the set of conditions and a set of internal and external status of the model.

In our model, a production rule is viewed as an associative relationship between a set of conditions and a subgoal as described in section 2.2. Storing production rules and retrieving a subgoal is represented within the framework of HASP, which can perform set intersection retrieving of a subgoal by providing relevant conditions simultaneously, without making a serial search (Hirai, 1983).

A subgoal, however, is only a higher level language, so that each subgoal must be broken down into a sequence of micro-procedures. Table 2.1 shows subgoals and micro-procedures postulated in this work. Each micro-procedure sends a set of control signals that control the behavior of each part of the model. Control signals constituting micro-procedures are shown in Table 2.2. A micro-procedure composed of control signals $CS-1$ and $CS-2$, $MP-1$, for example, controls the model to read the operator "+" from the external world and to store it in the working memory.

In order to reflect these hierarchical relationships between subgoals and micro-procedures, the procedural memory is devided into two subsystems as shown in Fig. 2.3-a. As shown in Fig. 2.3-b, a subgoal transition net (SGi, etc) is stored in the superordinate procedural memory, and sequences of micro-procedures ($MP-i$, etc.), which constitute subgoals, are stored in the subordinate procedural memory. According to the status of the model, a subgoal is retrieved from the superordinate subsystem. The subgoal specifies which sequence of micro-procedures the subordinate system should retrieve. The transition between micro-procedures is spontaneously initiated by the change in status signals resulting from the execution of the previous micro-procedure. When all micro-procedures constituting a subgoal are executed, the subordinate subsystem sends "Return" information concerning the completion of this subgoal to the superordinate procedural memory. According to "Return" information and the other new status, the next subgoal that should be executed will be retrieved.

Table 2.1. Subgoals and micro-procedures

SUBGOAL-1: Start
MP-1: Read operator from paper and temporarily associate control signal P_0 with it in the working memory.
SUBGOAL-2: Read-first-digit
MP-2: Read first digit from paper and temporarily associate control signal P_1 with it in the working memory.
SUBGOAL-3: Read-next-digit
MP-3: Read next digit from paper and temporarily associate control signal P_2 with it in the working memory.
SUBGOAL-4: Add-two-digits
MP-4.1: Reset input and output buffers of the model.
MP-4.2: Activate control signal P_0 and retrieve an operator from the working memory.
MP-4.3: Put the operator into the operator input buffer.
MP-4.4: Activate control signal P_1 and retrieve an augend form the working memory.
MP-4.5: Put the augend into the augend input buffer.
MP-4.6: Activate control signal P_2 and retrieve an addend from the working memory.
MP-4.7: Put the addend into the addend input buffer.
MP-4.8: Associate control signal P_1 with an answer, obtained from the semantic memory, in the working memory.
MP-4.9: Associate control signal P_{C1} with a carry, obtained from the semantic memory, in the working memory.
SUBGOAL-5: Write-carry
MP-5: Write a carry on paper for later use.
SUBGOAL-6: Add-carries
MP-6.1: Reset input and output buffers of the model.
MP-6.2: Activate control signal P_0 and retrieve an operator from the working memory.
MP-6.3: Put the operator into the operator input buffer.
MP-6.4: Activate control signal P_{C1} and retrieve a current carry from the working memory.
MP-6.5: Put the carry into the augend input buffer.
MP-6.6: Read the carry most recently written on paper and associate control signal P_{C2} with it in the working memory.
MP-6.7: Put the addend into the addend input buffer.
MP-6.8: Write the answer obtained from the semantic memory on paper as an accumulated carry.
SUBGOAL-7: End-of-column
MP-7.1: Activate control signal P_1 to retrieve a digit from the working memory and put the digit into the output buffer.
MP-7.2: Write the digit in the answer part of the current column.
MP-7.3: Read the carry most recently written and associate control signal P_1 with it in the working memory.
MP-7.4: Shift attention to the next column.
SUBGOAL-8: End-with-carry
MP-8.1: Move a carry from the previous column to the answer part of the current column.
MP-8.2: End.
SUBGOAL-9: End-without-carry
MP-9: End.

Table 2.2. Control signals constituting micro-procedures

CS-1:	Read a digit or an operator from paper and put it into the working memory.
CS-2:	Direct attention to an operator on paper.
CS-3:	Direct attention to the first digit on paper.
CS-4:	Direct attention to the next digit on paper.
CS-5:	Direct attention to a carry on paper.
CS-6:	Write a digit in the answer part of the current column.
CS-7:	Write a digit in the carry part of the current column.
CS-8:	Move the carry of the previous column to the answer part of the current column.
CS-9:	Write the answer obtained from the semantic memory on paper.
CS-10:	Write the carry obtained from the semantic memory on paper.
CS-11:	Write the digit retrieved from the working memory on paper.
CS-12:	Enable following control signals to the working memory.
CS-13:	Control signal P_0 to the working memory.
CS-14:	Control signal P_1 to the working memory.
CS-15:	Control signal P_2 to the working memory.
CS-16:	Control signal P_{C1} to the working memory.
CS-17:	Control signal P_{C2} to the working memory.
CS-18:	Put the answer obtained from the semantic memory into the working memory.
CS-19:	Put the carry obtained from the semantic memory into the working memory.
CS-20:	Put the digit or operator retrieved from the working memory into one of the input buffers of the semantic memory.
CS-21:	Reset input and output buffer of the model.
CS-22:	Shift attention to the next column.
CS-23:	End the processing.
CS-24:	Direct attention to a digit of the right column.
CS-25:	Move the carry of the current column to the answer part.
CS-26:	SUBGOAL-1 has been completed.
CS-27:	SUBGOAL-2 has been completed.
CS-28:	SUBGOAL-3 has been completed.
CS-29:	SUBGOAL-4 has been completed.
CS-30:	SUBGOAL-5 has been completed.
CS-31:	SUBGOAL-6 has been completed.
CS-32:	SUBGOAL-7 has been completed.
CS-33:	SUBGOAL-8 has been completed.
CS-34:	SUBGOAL-9 has been completed.
CS-35:	SUBGOAL-10 has been completed.

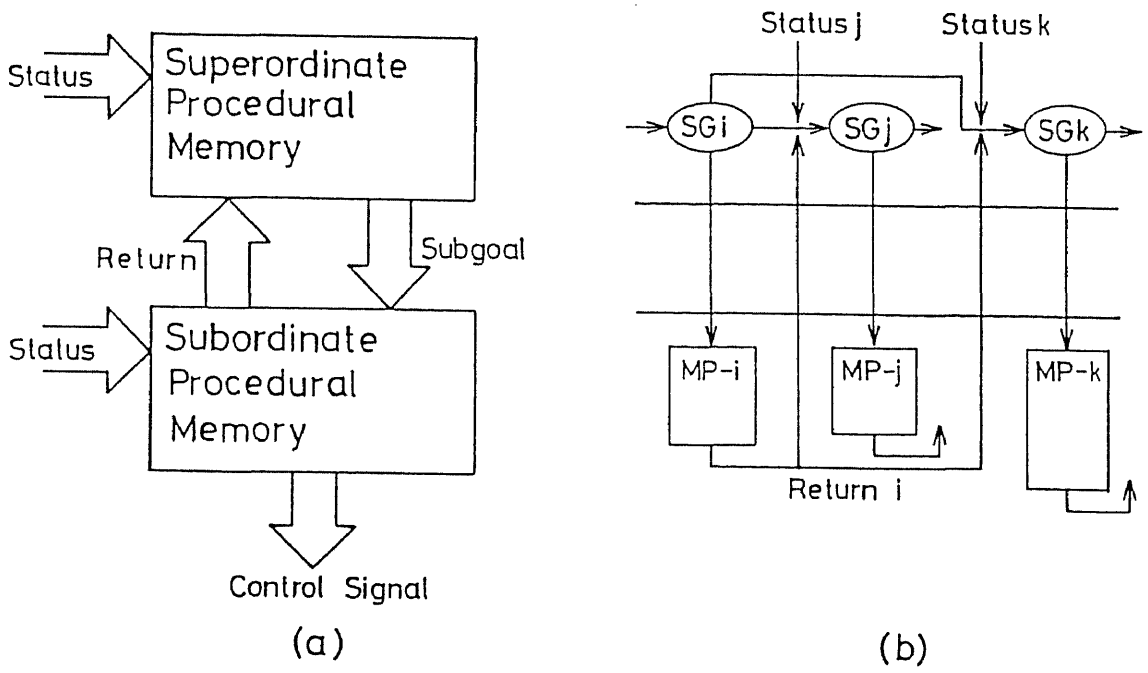


Fig. 2.3. Structure of the procedural memory. (a)The procedural memory is divided into two hierarchical subsystem: a superordinate procedural memory and a subordinate procedural memory. (b)Relation between subgoal transition net, denoted by SGi etc. in the superordinate procedural memory, and micro-procedures, denoted by MP-i etc. in the subordinate procedural memory.

2.4.1 Superordinate procedural memory

Structure of the superordinate procedural memory is shown in Fig. 2.4. It consists of HASP with positive feedback shown in Fig. 2.1 and a contextual filter to be described below. The subgoal transition net stored in the superordinate subsystem is shown in Fig. 2.5. In this figure, the notations $SGi's$ express subgoals, and $C_j's$ express the status signals that control subgoal transitions. For example, when $SUBGOAL-2$ is completed, if status C_3 exists, $SUBGOAL-9$ will be executed next. Table 2.3 shows these status signals. To describe how this transition net is stored in the superordinate procedural memory and how the next subgoal is retrieved, let us consider the case of $(SG2, C_3, SG9)$. This transition means: $SUBGOAL-2$ is maintained until it is achieved; When $SUBGOAL-2$ is achieved, which is informed by the subordinate procedural memory, and if status C_3 exists, $SUBGOAL-9$ is retrieved. To store this transition net, at first the pattern for $SUBGOAL-2$ is presented through $T(y_1)$ and $A(y_2)$ shown in Fig. 2.1 at the same time, and the associative relationship to itself is established in connections $W_{SA}(y_1, y_2)$ shown in Fig. 2.1 by strengthening their strength according to the learning algorithm $(L-2.2)$. Next, the associative relationship between a set of conditions, which consists of the status C_3 and the "Return" signal informing the completion of $SUBGOAL-2$, and $SUBGOAL-9$, and the serial associative relationship from $SUBGOAL-2$ to $SUBGOAL-9$ are stored in $W_{SK}(y_1, x)$ and $W_{SA}(y_1, y_2)$ respectively, by presenting the pattern for the set of conditions through $K(x)$, presenting the pattern for $SUBGOAL-2$ through $A(y_2)$, and presenting the pattern for $SUBGOAL-9$ through $T(y_1)$ simultaneously. It should be noted

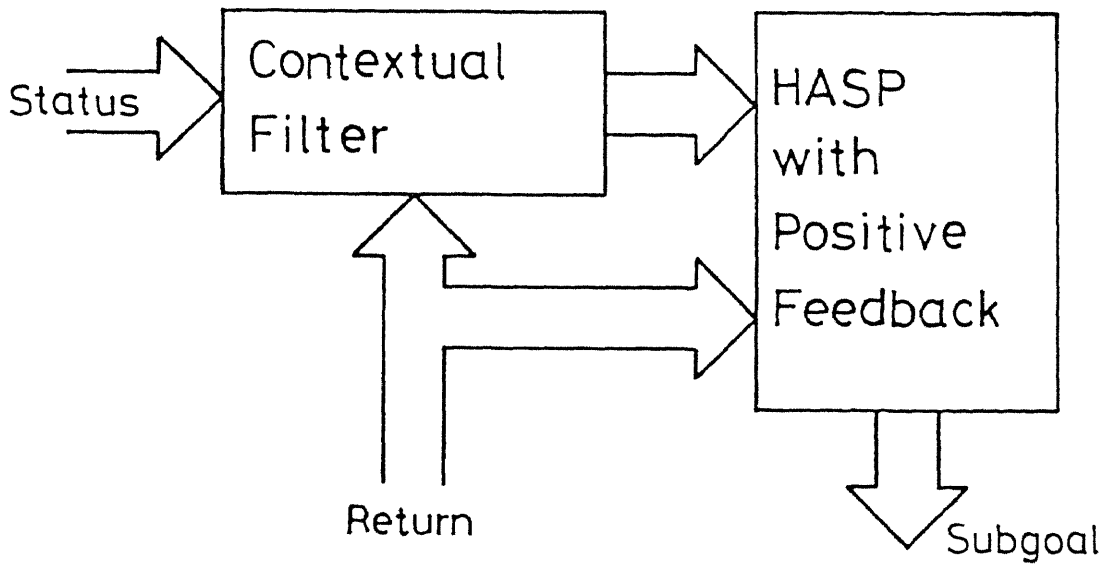


Fig. 2.4. Structure of the superordinate procedural memory. It consists of a contextual filter and HASP with positive feedback.

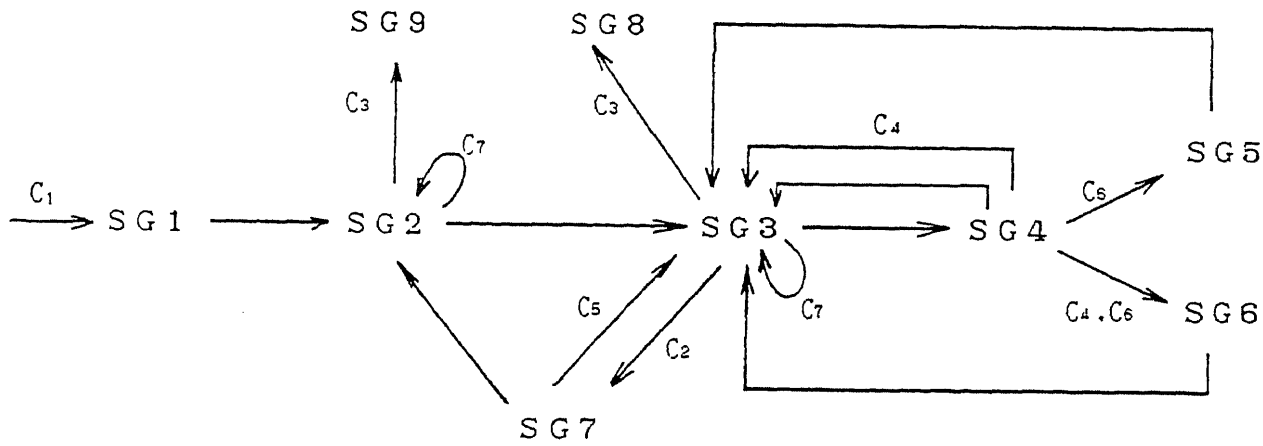


Fig. 2.5. Subgoal transition net stored in the superordinate procedural memory. SG1, SG2, ..., and SG9 indicate the subgoals and C_1, C_2, ..., and C_7 indicate the status signals controlling the subgoal transitions.

Table 2.3. Status signals controlling the subgoal transition

C-1:	Start of addition.
C-2:	End of a column.
C-3:	All digits have been processed.
C-4:	There is a carry of the current column on paper.
C-5:	There is a carry of the previous column on paper.
C-6:	There is a carry in the output of the semantic memory.
C-7:	Blank figure is read

that in order to retrieve $SUBGOAL-9$, the above serial associative relationship is also necessary, because $SUBGOAL-2$ will not disappear and is fed to the network through the positive feedback loops until $SUBGOAL-9$ is retrieved. Therefore, the stored associative relationships for the transition described above can be expressed as follows:

$$M[SG-2, f(SG-2)] = \{SG-2\}, \tag{2.8}$$

and

$$M[C_3, \text{Return of } SG-2, SG-2,$$

$$f(C_3, \text{Return of } SG-2, SG-2)] = \{SG-9\}. \tag{2.9}$$

The $SUBGOAL-2$ can be maintained by (2.8) until its completion. When $SUBGOAL-2$ is completed, a return signal that informs the completion of the subgoal is provided from the subordinate procedural memory. At this point, if the status C_3 exists, $SUBGOAL-9$ will be retrieved by (2.9).

Subgoal transition is controlled by the status signals as shown in Fig. 2.5. However, there is a case where unnecessary status signals exist together with necessary one for a transition. For example, when $SUBGOAL-3$ is completed, the status signals C_4, C_5 and C_6, which may exist, are irrelevant to the transition. These unnecessary status signals will disturb correct transition. To avoid this problem, a contextual filter is introduced. It is used to filter out status signals irrelevant to the transitions of the subgoals.

Structure of the contextual filter is shown in Fig. 2.6. As shown in the figure, the status signals are supplied through $Status(y)$, and the context signals which

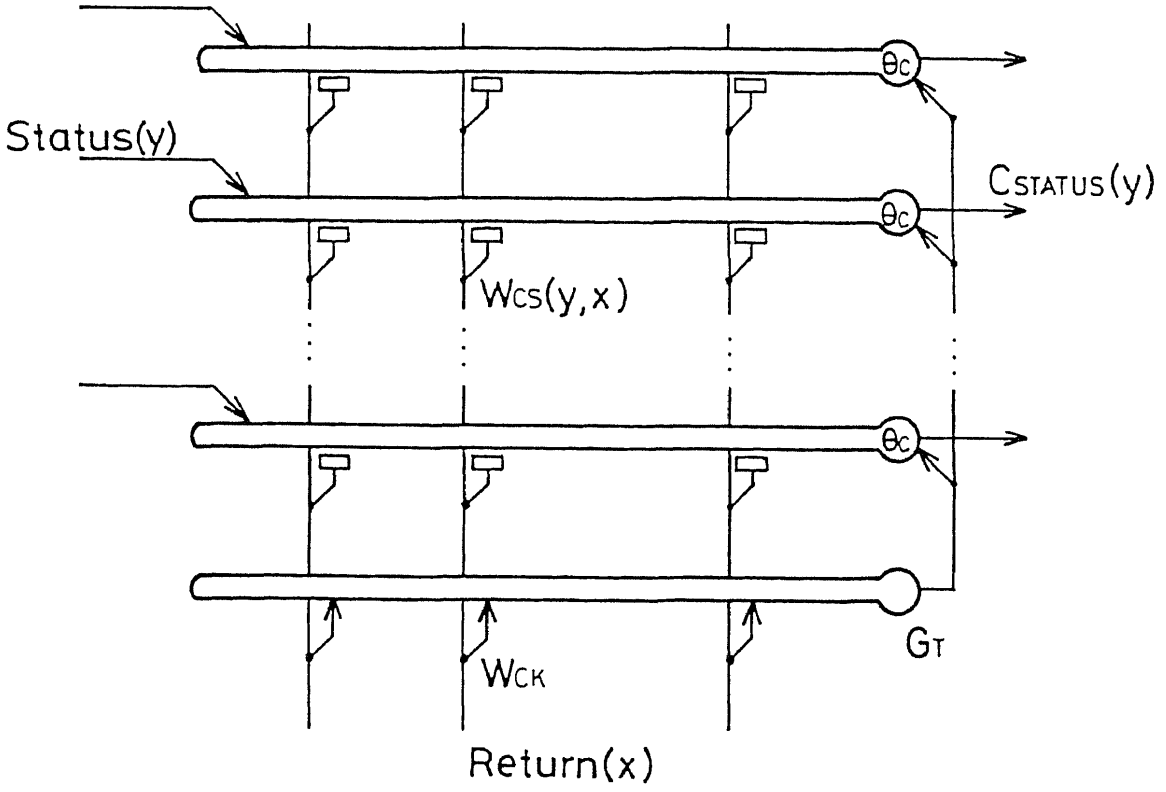


Fig. 2.6. Structure of the contextual filter. The network notations are the same as those in Fig. 2.1.

are the return signals informing the completion of subgoal are supplied through $Return(x)$. The output of the filter is defined by

$$C_{STATUS}(y,t) = \varphi[Status(y,t) - \sum_x W_{CS}(y,x) \cdot Return(x,t) + G_T - \theta_C], \quad (2.10)$$

and

$$G_T = \sum_x W_{CK} \cdot Return(x). \quad (2.11)$$

The inhibitory connections denoted by $W_{CS}(y,x)$ are modifiable and their strength is reduced from initial positive value W_{CS} to zero according to the following learning algorithm (L-2.5):

1. If $Status(y) \cdot Return(x) > \theta_L$, $W_{CS}(y,x)$ is reduced to zero,

2. Otherwise, $W_{CS}(y,x)$ is not changed.

In order to avoid undesirable modification of $W_{CS}(y,x)$, the magnitudes of $Status(y)$ and $Return(x)$ in a learning phase are assumed to be larger than those in normal operation. If a status signal $Status(y)$ is irrelevant to the transition initiated by $Return(x)$, the inhibitory connection $W_{CS}(y,x)$ is not reduced to zero. Hence, this statau signal is filtered out by (2.10), provided that $W_{CS} \geq Status(y,t) + G_T - \theta_C$. Oppositely, if a status is relevant to the transition, the corresponding inhibitory connection is reduced to zero. Hence, the status can be passed through the network by (2.10). In order to maintain a subgoal by itself through the positive feedback loops until its completion, status signals to superordinate memory must be blocked out in the meantime. This is carried out by the threshold θ_C in (2.10). Hence, θ_C must be not less than $Status(y,t)$. The element G_T is introduced to pass relevant

status signals only when some return signal is activated. Hence, G_T must be not less than θ_C. That is, W_{CK} must be not less than θ_C by (2.10), provided that the magnitude of $Return(x)$ is 1.0. Therefore, the network parameters must have the following relationships: $Status(y,t) \leq \theta_C \leq W_{CK} < W_{CS}$.

2.4.2 Subordinate procedural memory

The subordinate procedural memory is constructed by HASP with positive feedback. It can store and retrieve sequences of micro-procedures constituting subgoals. The status signals controlling the transition from one micro-procedure to next one are shown in Table 2.4. To describe how the sequences of micro-procedures are stored and retrieved in the memory system, let us consider the sequence of micro-procedure constituting $SUBGOAL-4$ as an example (see Table 2.1). In order to retrieve the first micro-procedure of $SUBGOAL-4$, at first the following associative relationship must be established:

$$M[SG-4, f(SG-4)] = \{MP-4.1\}. \tag{2.12}$$

In order to maintain the micro-procedure until its completion, the associative relationship to itself must also be stored through positive feedback as follows:

$$M[SG-4, MP-4.1, f(SG-4, MP-4.1)] = \{MP-4.1\}. \tag{2.13}$$

In order to retrieve the next micro-procedure $MP-4.2$ after the completion of $MP-4.1$, the associative relationship between the status $MC-2$, which results from the execution of the previous micro-procedure $MP-4.1$, and $MP-4.2$ must be established as follows:

$$M[SG-4, MC-2, MP-4.1, f(SG-4, MC-2, MP-4.1)]$$

Table 2.4. Status signals controlling the micro-procedure transition

MC-1:	Operation of the working memory has been completed.
MC-2:	Input and output buffers have been cleared.
MC-3:	Attention has been shifted to the next column.
MC-4:	Operation of the output subsystem has been completed.
MC-5:	A carry has been moved to the answer part of the current column.
MC-6:	A digit or an operator has been set in one of the input buffers of the semantic memory

$$= \{MP - 4.2\}. \tag{2.14}$$

In order to send return signal informing the completion of $SUBGOAL - 4$ after the completion of the last micro-procedure $MP - 4.9$, the associative relationship

$$M[SG - 4, MC - 1, MP - 4.9, f(SG - 4, MC - 1, MP - 4.9)]$$

$$= \{\text{Return of } SG - 4, \} \tag{2.15}$$

must be stored. $MC - 1$ in the above equation is a status signal resulting from the execution of $MP - 4.9$.

2.5 Model of Working Memory

Working memory is defined as a memory system that temporarily stores information necessary to carry out a specific problem-solving task (Squire & Cohen, 1984). In our model, working memory is used to temporarily store a result and a carry of addition of two digits, as well as an aguend, an operator, and an addend which are read from visual field. All of the models of working memory so far proposed assumed that it is an independent memory system with a separate box. In this work, however, it is shown that the working memory does not necessarily have a separate box, and it can be modeled within the framework of HASP. Working memory is assumed to be a temporary associative relationship between a specific control signal in a micro-procedure and a numerical concept or an operator activated by that procedure. For example, when the micro-procedure $MP - 2$ (see Table 2.1) is executed, the control signal $CS - 14$ shown in Table 2.2, which is embedded in $MP - 2$, is used to associate with a digit read from paper according to this micro-procedure.

Structure of the model of the working memory is shown in Fig. 2.7. The following two points are different from HASP:

1. The elements $S(y)$ and $A(y)$ in HASP are integrated into a single element denoted by $A_{CON}(y)$.

2. The excitatory connection denoted by $W_{WORK}(y,x)$ is a high speed modifiable one. It is strengthened to W by the conjunctive activation of $K_{DC}(x)$, which is a control signal, and $A_{CON}(y)$, which represents a numerical concept or an operator, but it is weakened as time goes.

The element $A_{CON}(y)$ receives numerical concept or operator input from the input encoding system or the semantic memory through $T_{CON}(y)$. The input to element $A_{CON}(y)$ through $T_{CON}(y)$ is controlled by control signals coming from the procedural memory ($CS-1$, $CS-18$, and $CS-19$ shown in Table 2.2) and can be expressed as follows:

$$T_{CON}(y,t) = A_{PM}(1,t) \cdot A_{INPUT}(y,t) + A_{PM}(18,t) \cdot A_{SMR}(y,t) + A_{PM}(19,t) \cdot A_{SMC}(y,t), \quad (2.16)$$

where, $A_{PM}(i,t)$ represents the response of ith output element of the subordinate procedural memory, from which control signal $CS-i$ shown in Table 2.2 may be sent, at time t. $A_{INPUT}(y,t)$ represents the response of an output element of input encoding system representing a nemerical concept or an operator at time t. $A_{SMR}(y,t)$ represents the response of an output element of the semantic memory, from which an intermediate result of addition comes, at time t. $A_{SMC}(y,t)$ represents the response of an output element of the semantic memory, from which a

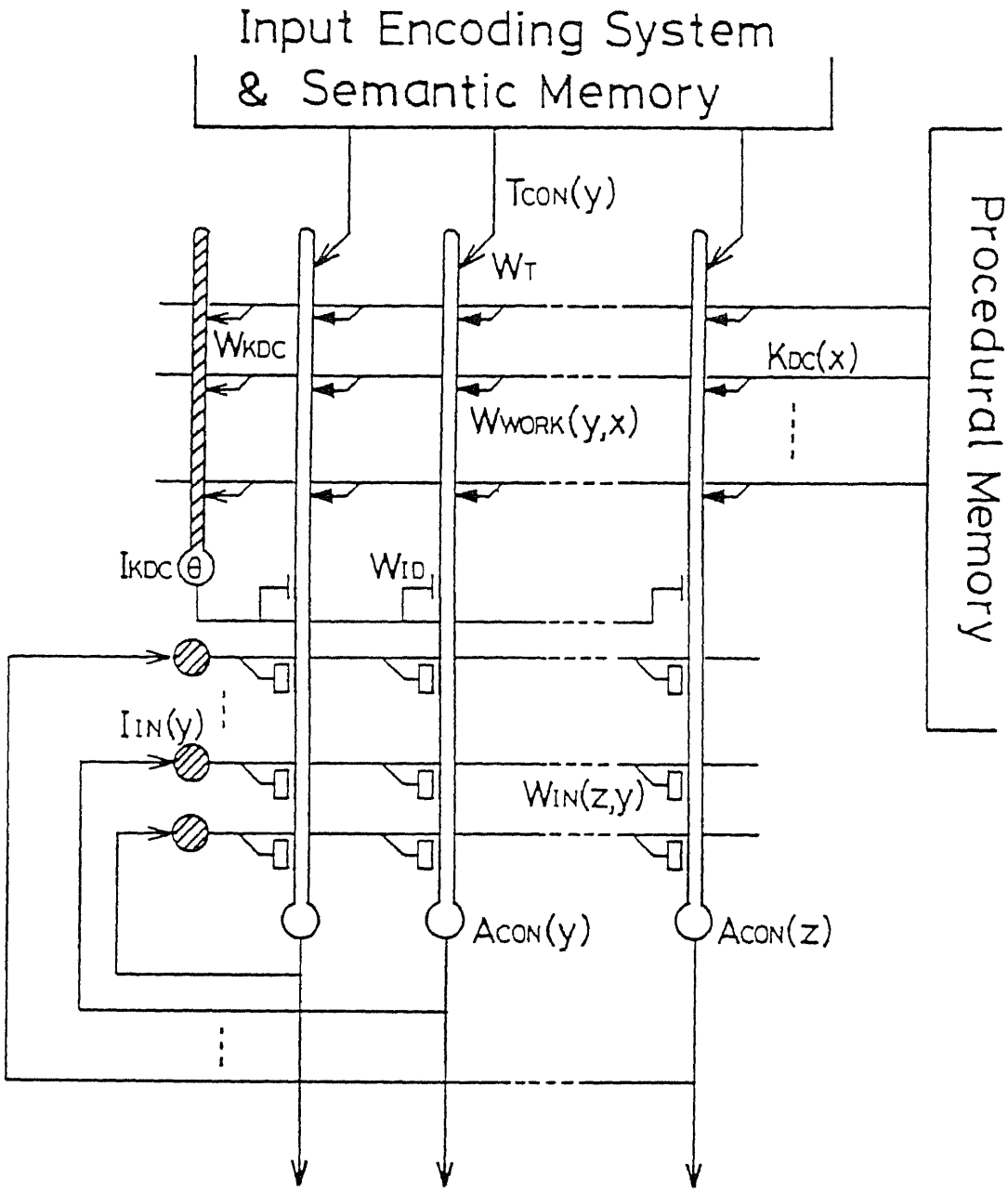


Fig. 2.7. Structure of the working memory. Network notation →: excitatory high-speed modifiable connection with decaying factor. The other network notations are the same as those in Fig. 2.1.

carry of addition comes, at time t. The key information supplied through $K_{DC}(x)$ consists of a set of control signals ($CS-12 \sim CS-17$) that comes from procedural memory as follows:

$$K_{DC}(x,t) = A_{PM}(12,t) \cdot A_{PM}(x+12,t), \tag{2.17}$$

where x takes an integer from 1 to 5. This equation means that the key input $A_{PM}(x+12,t)$ $(x=1,\ldots,5)$ is gated by $A_{PM}(12,t)$ at time t.

The element $A_{CON}(y)$ is assumed to be an analog threshold element with lag of the first order. the response is defined by

$$\mu_3 \frac{dA^*_{CON}(y,t)}{dt} + A^*_{CON}(y,t) = \sum_x W_{WORK}(y,x,t) \cdot K_{DC}(x,t)$$

$$- W_{ID} \cdot I_{KDC}(t) - \sum_z W_{IN}(y,z,t) \cdot I_{IN}(z,t) + W_T \cdot T_{CON}(y,t), \tag{2.18}$$

$$I_{KDC} = \sum_x W_{KDC} \cdot K_{DC}(x) - \theta, \tag{2.19}$$

$$A_{CON}(y,t) = \varphi[A^*_{CON}(y,t)], \tag{2.20}$$

and

$$I_{IN}(z,t) = A_{CON}(z,t). \tag{2.21}$$

The initial strength of the connection $W_{WORK}(y,x)$ is zero and is temporarily modified by the following learning algorithm (L-2.6):

$$W_{WORK}(y,x,t+dt) = \varphi_W[b \cdot W_{WORK}(y,x,t) + c \cdot A_{CON}(y,t) \cdot K_{DC}(x,t)]. \tag{2.22}$$

Where b $(0 < b < 1)$ is a decaying coefficient and c is a strengthening coefficient. The function $\varphi_W[*]$ is defined by

$$\varphi_W[a] = \begin{cases} W, & \text{if } a \geq W \\ a, & \text{if } 0 < a < W \\ 0, & \text{if } a \leq 0. \end{cases} \tag{2.23}$$

The inhibitory recurrent network composed of $W_{IN}(z, y)$ behaves as a competitive network in the same way as the homologue of HASP. Since the excitatory connection denoted by $W_{WORK}(y, x)$ is weakened as time goes, the inhibitory recurrent network suppresses older associatiated concepts and the most recently associated one can be retrieved. Besides, each time a temporarily stored concept is retrieved, $W_{WORK}(y, x, t + dt)$ is increased since $A_{CON}(y, t) \cdot K_{DC}(x, t) > 0$, and the memory for this concept is restrengthened. Therefore, when the model performs addition, once the operator "+" is registered in the working memory, it can be retrieved when necessary, without seeing it again.

2.6 Model of Semantic Memory

The primitive knowledge of addition, from $0 + 0 = 0$ to $9 + 9 = 18$, is stored in the semantic memory. If the patterns for "6", "+", and "8" are supplied to the semantic memory in parallel, for example, the patterns for answer "4" and carry "1" can be retrieved simultaneously. Such a semantic memory can be modeled by using an orthogonalizing network proposed by Hirai (see Hirai & Ma, 1988) and a heteroassociative network of HASP.

As mention above, HASP retrieves associative items by set intersection operation. By storing all associative relationships of addition from $0+0 = 0$ to $9+9 = 18$,

all of the augend and addend come to be associated with multiple numbers, and the right answer can not be retrieved by the intersection of sets specified by an augend and an addend. To solve this problem, the orthogonalizing network is employed, and the all key input patterns, from $0+0$ to $9+9$, are transformed into non-overlapping ones.

Structure of the model is shown in Fig. 2.8. The key input is divided into three parts K_{AUG}, K_O, and K_{ADD}, through which the patterns for augend, operator, and addend are inputted respectively. The output is divided into two parts A_{SMR} and A_{SMC}, from which the patterns for intermediate result and carry come out simultaneously. In order to store $6+8=14$, for example, the patterns for"6", "+", and "8" are presented through K_{AUG}, K_O, and K_{ADD} as key inputs, and the patterns for "1" and "4" are presented as associative inputs to the heteroassociative network. Hence,the association $M[f(\text{“6”}, \text{“}+\text{”}, \text{“8”})] = \text{“14”}$ can be established. During the process of adding, by providing the patterns for "6", "+", and "8" in parallel, the result "4" and the carry "1" can be retrieved from A_{SMR} and A_{SMC} simultaneously.

It is assumed that the semantic memory has input buffers composed of neuron-like elements with positive feedback to maintain a key input until all keys of augend, operator, and addend are prepared. The input buffers are controlled by control signals from procedural memory as follows:

$$K_{AUG}(y,t+dt) = [(A_{PM}(20,t)\cdot(A_{PM}(14,t)+A_{PM}(16,t))\cdot A_{CON}(y,t) + K_{AUG}(y,t)]\cdot\varphi[1-A_{PM}(21,t)], \tag{2.24}$$

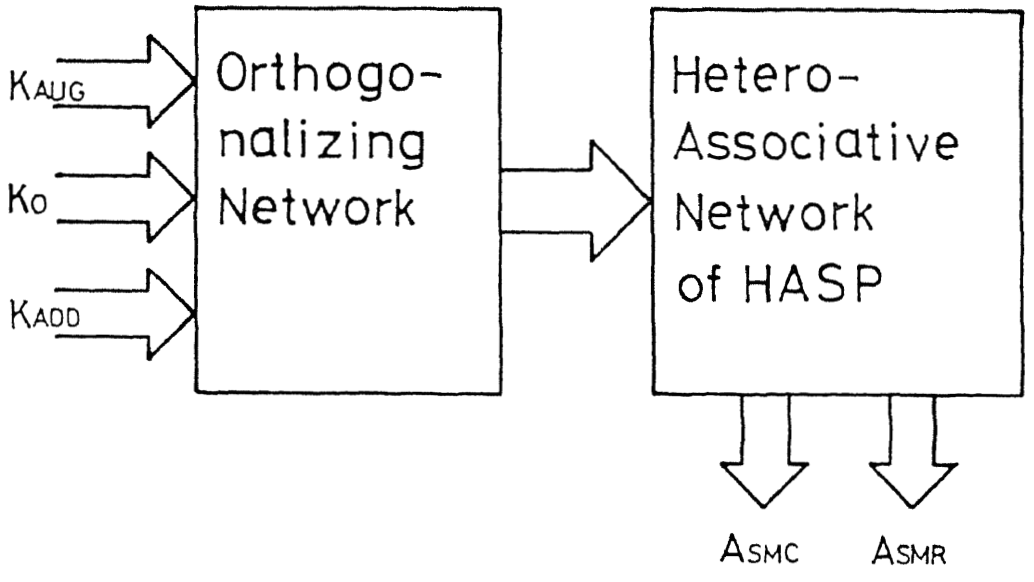


Fig. 2.8. Structure of the semantic memory.

$$K_O(y,t+dt) = [A_{PM}(13,t) \cdot A_{CON}(y,t) + K_O(y,t)] \cdot \varphi[1 - A_{PM}(21,t)], \quad (2.25)$$

$$K_{ADD}(y,t+dt) = [A_{PM}(20,t) \cdot (A_{PM}(15,t) + A_{PM}(17,t)) \cdot A_{CON}(y,t)$$

$$+ K_{ADD}(y,t)] \cdot \varphi[1 - A_{PM}(21,t)], \quad (2.26)$$

where $A_{PM}(21,t)$ is a control signal that resets the input buffers.

2.7 Results of Simulation

The performance of the model was simulated on a digital computer. The patterns representing (encoding) digits from "0" to "9" and operator "+" are assumed to be arrays composed of 11 elements as follows: "0" = $(1,0,\ldots,0)$, "1" = $(0,1,0,\ldots,0)$,..., "9" = $(0,\ldots,0,1,0)$, "+" = $(0,\ldots,0,1)$. The network parameters used are as follows.

(1) Procedural memory (Fig. 2.1): The time constants of S- and A-elements, μ_1 and μ_2, of the superordinate procedural memory are set to 1.0 time units of numerical calculation. The time constants in the subordinate procedural memory are set to 2.0 time units of numerical simulations. The other network parameters in both the superordinate and subordinate procedural memory are the same and set as follows. W_K, W_I, and θ are set to 2.0, 1.0 and 1.0 respectively. The strength of the modifiable excitatory connections $W_{SK}(y_1,x)$, $W_{SA}(y_1,y_2)$ and $W_{SF}(y_1,z)$ are increased from 0 to 2.0 by learning. In order to avoid a state of mutual depression among competing items, the connecting coefficient $W(y_1)$ from $S(y_1)$ to $A(y_1)$ is

set randomly to a value between 1.0 and 1.2. The initial strength of the recurrent inhibition $W_{AA}(y_1, y_2)$ is set to 1.5 and reduced to 0 by learning.

(2) Contextual filter (Fig. 2.6): The excitatory connection W_{CK} is set to 1.0. The initial strength of the modifiable inhibitory connection $W_{CS}(y, x)$ is set to 20.0 and reduced to 0 by learning. The element threshold θ is set to 1.0. The learning threshold θ_L (see learning algorithm (L-2.5)) is set to 4.0.

(3) Semantic Memory: In this network S-elements are assumed to have no time lag. All the other network parameters are the same as those of the superordinate procedural memory.

(4) Orthogonalizing network: Initial value of the modifiable connections $PW_{SK}(y, x)$ is set randomly to a value between 0.0001 and 0.0005, and is increased to 20.0 by learning. The coefficient γ used to fix the threshold $P\theta$ is set to 0.99. (see Hirai & Ma, 1988 in detail).

(5)Working memory (Fig. 2.7): The time constant μ_3 of A_{CON}- elements is set to 2.0 time units of numerical simulations. Network parameters W_{KDC}, W_{ID} and θ are all set to 1.0. W_T is set to 4.0. Decaying cofficient b and strengthening coefficient c in equation (2.22) are set to 0.99 and 0.2 respectively. The magnitude of saturation of $W_{WORK}(y, x)$ is set to 2.0. The initial strength of the recurrent inhibition $W_{IN}(y, z)$ is set to 1.5 and reduced to 0 by learning.

By storing the two kinds of procedural knowledge, subgoal transition net shown in Fig. 2.5 and micro-procedures in Table 2.1, in the procedural memory and by storing the primitive knowledge of addition of two digits from 0+0=0 to 9+9=18 in the semantic memory, addition of multiple numbers with multiple digits can be per-

formed by the model. The simulation result of the process of adding 537+618+429, for example, is shown in Fig. 2.9-a. Long bars in the figure express the transition of subgoals and short bars express the transitions of micro-procedures. The intervals between short bars reflect the time length to complete micro-procedures, and the intervals between long bars reflect the time length to complete subgoals. The result shows that it took 1247 steps of numerical simulation to complete this addition.

2.7.1 Improvement in the efficiency of the performance

When we learn driving a car, for example, at first we drive deliberately under conscious control. Through practice, we can get to drive smoothly and efficiently, as if one driving action automatically primes the next one, and we sometimes integrete some driving procedural steps into one chunk.

This section shows that by making explicit serial associative relationships between successive micro-procedures, our model can merge the preceding procedure into the succeding one automatically, if the preceding one is a subset of the succeding one. Besides, the priming effect that facilitates the micro-procedural transitions can be obtained at almost every transition. Hence the performance of the model can be spontaneously improved.

Let us consider the case of $MP-4.2$ and $MP-4.3$ in the $SUBGOAL-4$ shown in Table 2.1. It should be noted that the active control signal in the $MP-4.2$ is only P_O. It must also be activated in the $MP-4.3$, because if P_O is not active, the operator retrieved by P_O will disappear from the the working memory before it is supplied to the semantic memory. Hence the $MP-4.2$ is a subset of the $MP-4.3$.

The micro-procedure currently executing is maintained in the subordinate pro-

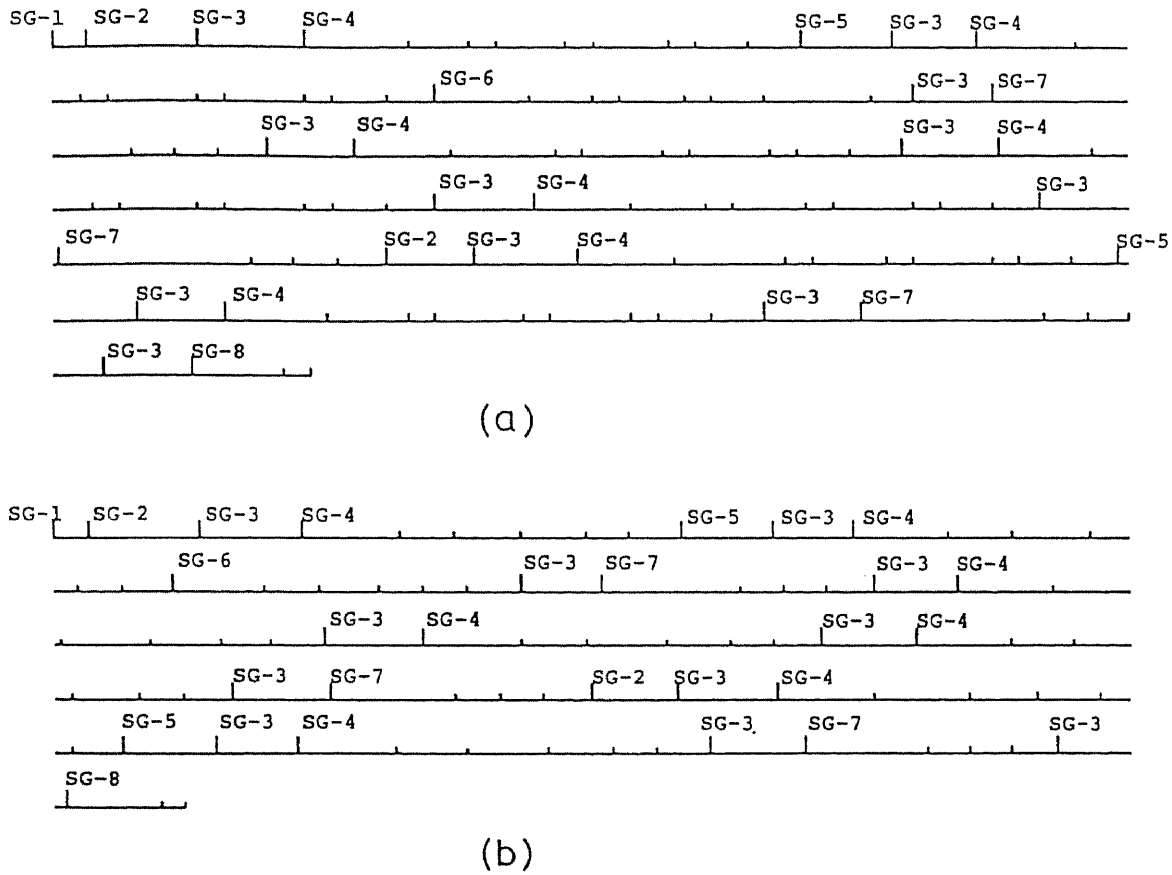


Fig. 2.9. The process of adding 537+618+429. (a)in the deliberate mode, and (b) in the skilled mode. Long bars indicate transitions between subgoals, and short bars indicate transitions between micro-procedures. It took 1247 steps (a) and 1023 steps (b) to complete the addition.

cedural memory by the positive feedback. Let the current micro-procedure and the subgoal be $MP-4.2$ and $SUBGOAL-4$ respectively. This self-maintenance associative relationships can be expressed as follows:

$$M[SG-4, MP-4.2, f(SG-4, MP-4.2)] = \{MP-4.2\}. \tag{2.27}$$

The transition to the next micro-procedure $MP-4.3$ is initiated by the status signal $MC-1$, which results from the execution of the current micro-procedure $MP-4.2$. The associative relationship between the status signal and the next micro-procedure can be expressed as follows:

$$M[SG-4, MC-1, MP-4.2, f(SG-4, MC-1, MP-4.2)]$$

$$= \{MP-4.3\}. \tag{2.28}$$

Since every transition of micro-procedures is executed one by one under the explicit control of status signals, this mode of operation can be called a deliberate mode. The process of adding 537+618+429 shown in Fig. 2.9-a is carried out in this mode.

If we make explicit serial associative relationship between current and suceeding micro-procedure, this serial association together with the self-maintenance association by (2.27) can be expressed as follows:

$$M[SG-4, MP-4.2, f(SG-4, MP-4.2)]$$

$$= \{MP-4.2, MP-4.3\}. \tag{2.29}$$

By (2.29), the next micro-procedure begins to appear at the output of the heteroassociative network of the subordinate procedural memory during the execution of a

current micro-procedure, and the priming effect can be obtained at almost every transition of micro-procedures. In addition, when a current micro-procedure is a subset of the succeeding one, the current micro-procedure is spontaneously merged into the succeeding one. For instance, in the case of $MP-4.2$ and $MP-4.3$, since $MP-4.2$ is a subset of $MP-4.3$, $MP-4.3$ can be retrieved directly by $SUBGOAL-4$ and $MP-4.2$ by (2.29)without waiting the completion of $MP-4.2$. It should be noted that in the deliberate mode, $MP-4.2$ can be maintained by $SUBGOAL-4$ and the current micro-procedure $MP-4.2$, because $f(SG-4, MP-4.2)$ uniquely specifies the $MP-4.2$ by (2.27). This kind of merging occurs between $MP-4.2$ and 4.3, $MP-4.4$ and 4.5, $MP-4.6$ and 4.7, $MP-6.2$ and 6.3, and $MP-6.4$ and 6.5. In Fig. 2.9-b, the process of adding 537+618+429 in this skilled mode is shown. It took 1023 steps to obtain the answer, and the performance could be improved about 20% compared with the deliberate mode. About fifty percent of the improvement comes from the priming effect, and the another fifty percent comes from merging of micro-procedures. In addition, it can be shown that more than two procedures can also be merged if preceeding procedures are subsets of succeding ones.

2.7.2 Generation of Bugs

By following incorrect procedures, children often make systematic errors in addition. This kind of error is called "bug". To analyze bugs provides a very important clue for understanding the mechanism of human cognitive processes. Several models on the generation and diagnosis of bugs have been proposed (Young & O'Shea, 1981; Brown & VanLehn, 1980).

By memorizing incorrect procedures, the model can generate four kinds of bugs observed in children's performance of addition (Cox, 1975). Let us consider one kind of bugs in detail. The bug is that children always barely add all digits without considering columns. In adding 476+17, for example, all digits are added as (4+7+6+1+7). We can generate this bug by storing the subgoal transition net shown in Fig. 2.10. The modifications of the subgoal transition net are as follows. After the completion of $SUBGOAL-3$, it is executed again without executing $SUBGOAL-7$, which writes out the answer of a column. Besides, if all digits have been added, $SUBGOAL-7$ is executed at first, and then $SUBGOAL-8$ or $SUBGOAL-9$ will be executed.

We can generate the other three kinds of bugs in the same way (Ma, 1987). By compairing the modifications made to the correct procedure to generate the four kinds of bugs, we found that most of them are related to the subgoals that should be retrieved after the addition of a column is finished or after all digits are processed. In other words, the bugs occur during inter-columnar processing ($SUBGOAL-7$) or after all digits are processed ($SUBGOAL-8$) and do not occur during intra-columnar processing ($SUBGOAL-4,5,6$). This may be a very natural finding, since column and carry are complicated concepts for children.

2.8 Summary

In this chapter a neural network model that can represent problem solving processes such as addition of multiple numbers with multiple digits has been described. In constructing the model, neural network models of semantic memory, procedural

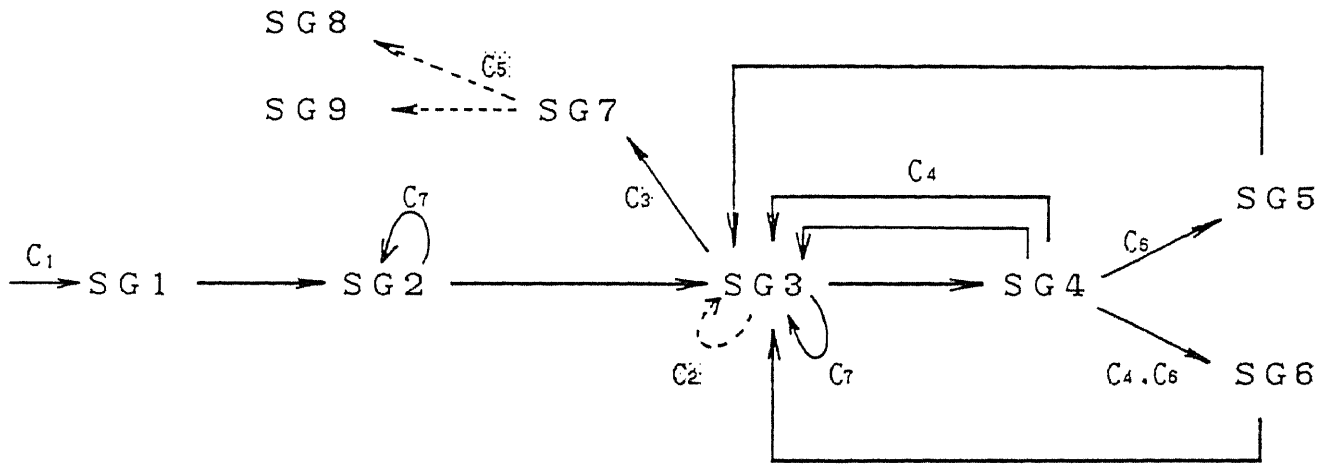


Fig. 2.10. The modified subgoal transition net for generating one kind of bugs. The shaded status signals and dotted transition lines are modified parts.

memory and working memory, which are necessary components to represent the process of adding, have been constructed within the framwork of HASP. By combining these memory models, it has been shown that the model can perform addition of multiple numbers with multiple digits. It has also been shown that the performance of the model can spontaneously be improved by introducing explicit serial associative relationships between consecutive procedures. By memorizing incorrect procedures, four kinds of bugs observed in children's performance can be represented. Since the proposed scheme has a general structure, it can easily be applied to various kinds of problem-solving tasks by introducing domain specific procedural and semantic knowledge.

Chapter 3

Knowledge Representation

In this chapter we propose a neural network model of semantic network, which can memorize inheritance hierarchies with exceptions, within the framework of HASP. The model consists of two networks: One can answer the "what" type and "yes-no" type of queries, and the other can answer recognition problems. Because the property inheritance is established by the associative relationships from the subordinate concepts to the superordinate ones, it is not necessary to associate all the properties the superordinate concepts have with the subordinate concepts. This will be very parsimonious in implementing the model in true hardware. It is also not necessary to introduce the property inheritance into the encoding scheme of concepts per se, as Hinton's model adopted (Hinton, 1981). In his model, the code of a subordinate concept must include the codes of all it's superordinate ones. Besides, the model can select a best answer in the recognition tasks, even when a query lacks or excesses some input properties. The results of simulation studies show that our model can perform query answering and recognition

tasks very quickly and effectively, inspite of the existence of exceptions to the property inheritance.

3.1 Motivation

It seems very difficult for the serial, computer-like models to deal with large bodies of knowledge. A possible reason of the difficulty is that some operations and algorithms necessary for these problems take too long time. However, many of such operations or algorithms can be handled vary easily by some parallel machines. One example is the intersection operation of large sets: On a serial machine, it takes time in propertional to the product of the numbers of the members in all sets; On some parallel machines such as our models to be described later or Fahlman's model (Fahlman, 1979), it takes only a few cycles, regardless of the set size.

Fahlman's model, called NETL, was the first attempt at using a massively parallel network of simple processing elements to representing semantic net. However, since each node in the semantic net exactly corresponds to a particular element (node) of NETL and the elements communicated with one another by propagating discrete messages called *markers*, NETL is incapable of supporting "best match" operation in recognition (Brachman, 1985; Fahlman, 1982). That is, in the case that there is no concept that possessed *all* of the observed features (properties), NETL cannot retrieve a concept that most closely exhibits all of the obeserved features. Moreover, the model cannot represent the exceptions to the property inheritance which is a basic characteristic of human knowledge representation. Finally, NETL did not fully utilize the potential for parallelism because the internode

communication depended on instructions issued by a central (serial) controller. In the model proposed by Hinton (1981), each node in the semantic net was encoded by a particular pattern of activity on a large assembly of units. The pattern of a superordinate concept is inherited to the subordinate ones as their partial patterns, so that the property inheritance can be ensured. However, the interferences caused by the exceptions make the system take longer time to settle, and sometimes the retrieved patterns are degraded in the final state. In the recent works, the retrieving capablity of semantic net have been considered (Fahlman et al., 1981; Brachman, 1982). Etherington and Reiter (1987) established a correspondence between the inheritance hierarchies with exceptions and the theories of default logic (Reiter, 1980). So far, however, there is no approach to concretely realize such correspondence with neural network models.

In this chapter, we propose a new model, which can represent the inheritance hierarchies with exceptions and improve several defects existing in the previous models. Such a model is constructed within the framework of HASP.

The structure of this chapter is as follows. In Section 3.2, we describe the property inheritance, a basic characteristic of human knowledge representation, and the inheritance hierarchies with exceptions. Section 3.3 shows the structure of the model and describes how it memorize the inheritance hierarchies with exceptions and how they search the memorized concepts to answer queries and recognition problems. In addition, the functions of selecting a suitable answer from multiple candidates, the "best match" operation, and the method of ensuring the property inheritance of our model is compared to the previous ones, and the problems ex-

isting in the previous ones are pointed out. Section 3.4 presents the results of a computer simulation and shows that our model can handle the inheritance hierarchies with exceptions very quickly and effectively.

3.2 Knowledge Representation and Property Inheritance

We human being can memorize a tremendous quantity and variety of knowledge and can access whatever knowledge we need very quickly and flexibly. However, much of the information we use in everyday life is not stored explicitly but must be deduced from other information (Fahlman, 1979). When we are told the fact that Clyde is an elephant, for example, we immediately know "What color he or she is", "Whether he or she has long nose", and so on. Obviously, we cannot imagine that these knowledge about Clyde are stored in the explicit form. Our alternative is to believe that we have such knowledge about elephant and we know that the properties the superordinate concepts (e.g. elephant) have are inherited to their subordinate ones (e.g. Clyde) in general. So, if we know in addition the facts "Every elephant is a mammal" and "Every mammal needs air", according to the property inheritance we can deduce "Elephant needs air", and can further deduce "Clyde needs air".

A real-world example of concept hierarchy net with properties (or called IS-A hierarchy with properties, or called inheritance hierarchy) is shown in Fig. 3.1. From this net, we know that African elephants are elephant, mammal, and animal and they have big ears and their color are relatively black. We can also deduce

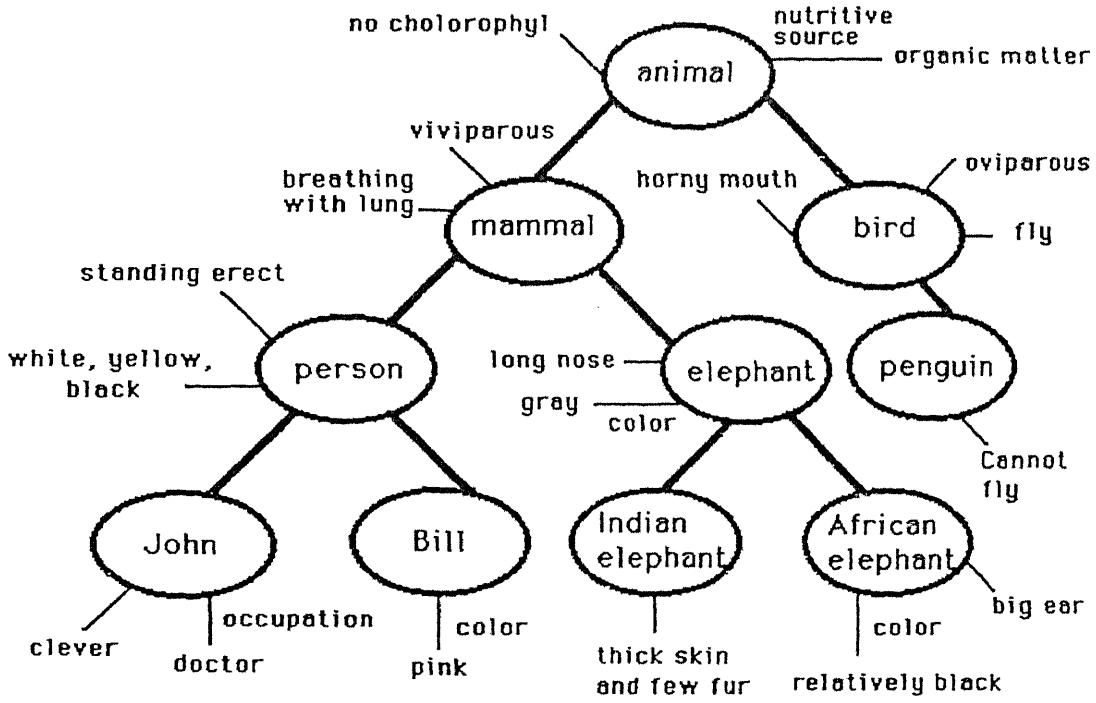


Fig. 3.1. A real-world example of inheritance hierarchy with exceptions.

that African elephants have long noses and that they breathe with lungs from the property inheritance, since the superordinate concept of African elephant, elephant, has long nose and the superordinate concept of elephant, mammal, breathes with a lung. On the other hand, exceptions to the property inheritance exist in the real-world knowledge. From this hierarchy, for example, we can see that penguin cannot fly, although the superordinate concept, bird, can fly in general.

From the above descriptions we can see that the property inheritance is a basic characteristic of human conceptual representation, and how to ensure it is a major issue in the works of representing semantic nets (Hinton, 1982). One method is to copy all the properties of superordinate concepts for every subordinate one. However, this straightforward approach is too space-consuming. A better alternative to making a full copy of the superordinate concept is to give each subordinate concept a pointer back to its superordinate one, so that whenever a question arises about the subordinate concept, the properties connecting with a superordinate one can be inspected.

3.3 The Model

3.3.1 A network for query answering

A part of the inheritance hierarchy with exceptions shown in Fig. 3.1 is shown in Fig. 3.2. In the figure, we refer to nodes such as "Bill" and "person" as concepts, labels such as "color" and "occupation" as properties of the concepts, the terminal descriptions associated with properties, such as "pink" and "doctor" as values of the properties, and the terminal descriptions directly associated with the concepts,

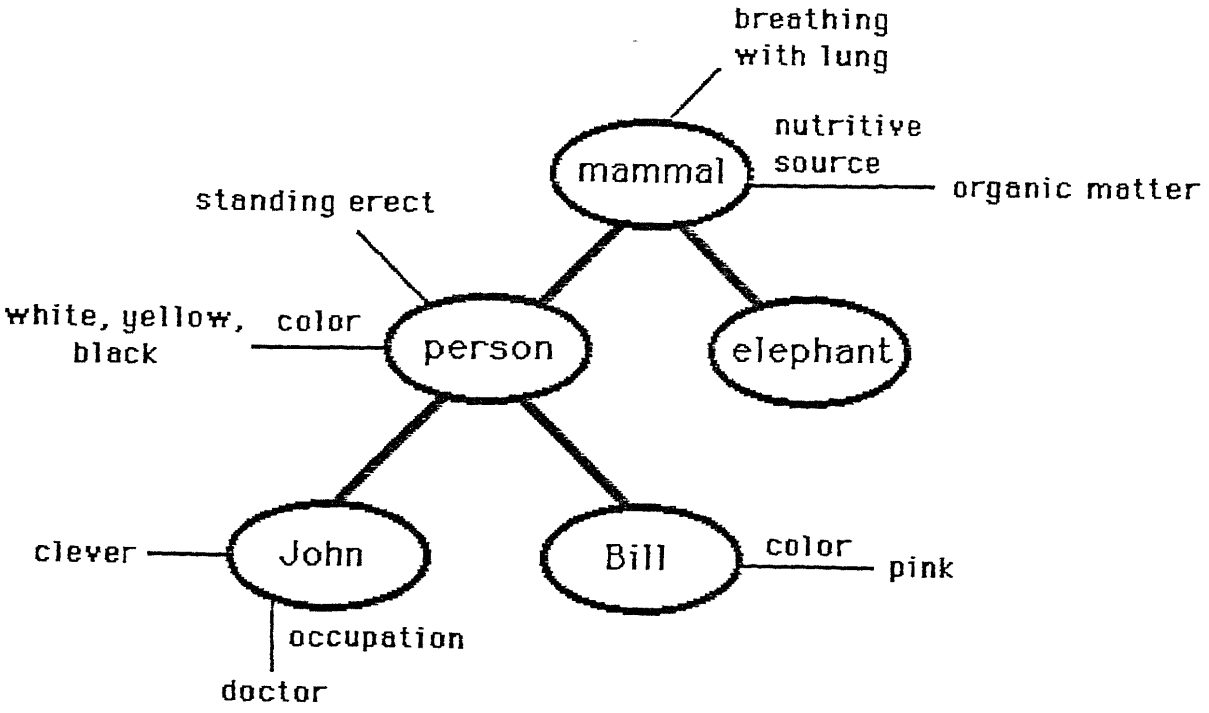


Fig. 3.2. A part of the inheritance hierarchy with exceptions shown in Fig. 3.1.

such as "clever" and "standing erect", as facts associated with the concepts. The information such as (Bill, color, pink) shown in Fig. 3.2 can be used to answer the "what type" query, "What color is Bill ?", and the information such as (person, standing erect) can be used to answer the "yes-no" type query, "Does person stand erect ?". In addition, as shown in the figure, the fact that the color of Bill is pink is an exception to the property inheritance, since the color of person, a superordinate concept of Bill, is white, yellow, or balck in common, not pink.

Structure of the network, which can memorize inheritance hierarchies with exceptions and can search the stored concepts to answer the two types of queries described above, is shown in Fig. 3.3. It consists of two associative networks: One, called IS-A network, is a heteroassociative network with positive feedback loops and store the associative relationships from the subordinate concepts to the superordinate ones; The other, called Property-Value network, consists of HASP without readout control units and store the associative relationships from the properties of concepts to their values.

Next, we describe how the model memorizes the inheritance hierarchy with exception shown in Fig. 3.2 and how it answers queries.

In IS-A network, by presenting each pattern for concept through $KN(x)$ and $TN(y_1)$ at the same time, the modifiable excitatory connections $W_{SK}(y_1, x)$ are strengthened according to the following learning algorithm (Hirai, 1983):

1. if $KN(x) \cdot TN(y_1) > 0$, $W_{SK}(y_1, x)$ is increased to some positve value W_S, and

2. if $KN(x) \cdot TN(y_1) = 0$, $W_{SK}(y_1, x)$ is not changed.

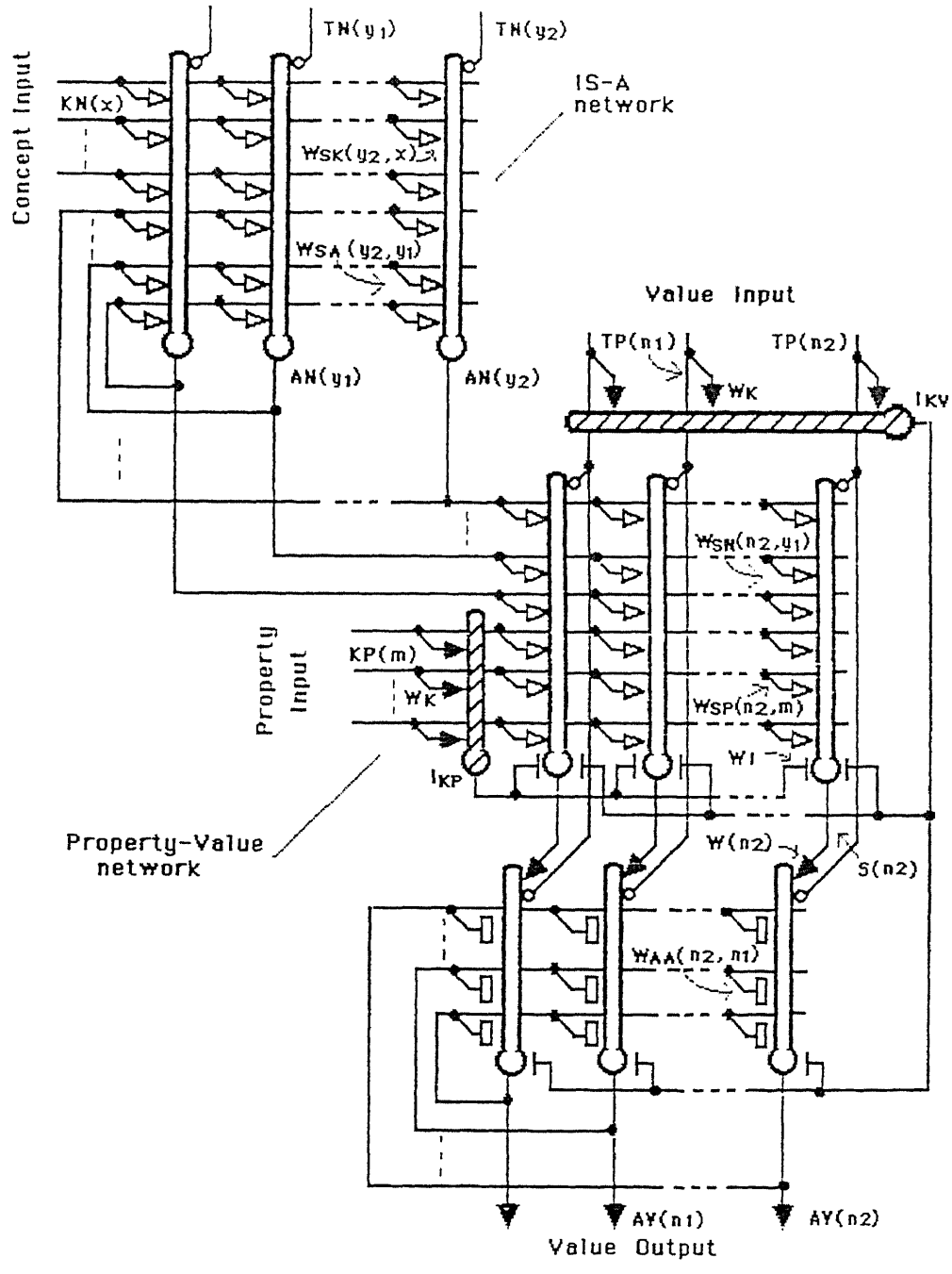


Fig. 3.3. Structure of the network for query-answering. Network notations are the same as those in Fig. 2.1.

Hence, the associative relationships of all concepts with themselves are established. By using an associative mapping function $M[*]$, these associative relationships can be expressed as follows: M["Bill"] = "Bill", ..., M["mammal"] = "mammal". Moreover, by presenting each pair of patterns for the subordinate concept and it's superordinate one through the positive feedback loops and $TN(y_1)$ respectively, all associative relationships from the subordinate concepts to the superordinate ones are stored in $W_{SA}(y_2, y_1)$. These associtive relationships are M["John"]="person", M["person"]="mammal", M["elephant"]="mammal", and M["Bill"]="person" (see Fig. 3.2). In Property-Value network, by presenting each pattern for concept through $AN(y_1)$, the pattern for the property of the concept through $KP(m)$, and the pattern for the value of the property or the fact associated with the concept through $TP(n_1)$, all associative relationships from the concepts and properties to their values or from the concepts to the associated facts explicitly shown in Fig. 3.2 are stored in $W_{SN}(n_2, y_1)$ and $W_{SP}(n_2, \mathrm{m})$. These associative relationships are M["John","occupation"]="doctor", M["John"]="clever", and so on. At the same time, the strength of the modifiable inhibitory connections $W_{AA}(n_2, n_1)$ are reduced from initial value to zero according to the learning algorithm $L-2.4$ described in Section 2.2.2 to store the autocorrelation of patterns for the values.

To answer a "what type" query, "What is the color of Bill ?", for example, the explicit information (Bill, color, pink), an exception to the propety inheritance as described above, shown in Fig. 3.2 can be used and the answer, "pink", for the query can be obtained easily by the model as follows: By presenting the

pattern for the concept "Bill" to IS-A network through $KN(x)$ and the pattern for the property "color" to Property-Value network through $KP(m)$, the pattern for "Bill" itself can be retrieved from IS-A network by M["Bill"]="Bill" stored in $W_{SK}(y_2, x)$ first. Concept "Bill" will activate it's superordinate one "person" and "person" will activate it's superordinate one "mammal" further through the positive feedback loops since the all associative relationships from the subordinate concepts to the superordinate ones are stored in $W_{SA}(y_2, y_1)$ of IS-A network. Finally a superimposed pattern of "Bill", "person", "mammal" will appear at the output of IS-A network and will be supplied to Property-Value network through $AN(y_1)$. Since the property "color" is being presented to Property-Value network through KP(m) at the same time, the values "pink", "white", "yellow", and "black" will appear at the output of the heteroassociative network of Property-Value network as a superimposed pattern by M["Bill","color"]="pink" and M["person","color"]={"white","yellow","black"}. The values not associated with the property "color" will not appear at the output of the heteroassociative network, since the inhibitory element I_{KP} is introduced and it's parameters are chosen so that it can completely suppress the activation of them. Finally, the pattern for "pink", the answer for this query, will win the cometition through the recurrent network and appear at the output of Property-Value network, since the retrieval of "Bill" which is associated with "pink" from IS-A network precedes that of "person" which is associated with the other three values as described above. Moreover, for a query "What is the nutritive source for John ?", for example, although no explicit information that can be used to directly answer the query is

stored in the model (see Fig. 3.2), the answer can also be obtained easily by the model in the same method: While the pattern for concept "John" being presented to IS-A network and the pattern for property "nutritive source" being presented to Property-Value network, the superimposed pattern of "John", "person", and "mammal" will appear at the output of IS-A network as described above and will be fed to Property-Value network, and the pattern for the value "organic matter", which is the answer for the query, can be retrieved from Property-Value network by M["mammal","nutritive source"]="organic matter" (see Fig. 3.2). It should be noted that no pattern will be retrieved from the network until the pattern for concept "mammal" appears at the output of IS-A network because of the inhibitory element I_{KP}. Moreover, for the queries such as "What is the occupation of Bill ?", since no knowledge that can be used to answer such queries is stored in the model, no groundless pattern will be retrieved.

To answer a "yes-no" type query "Does John breathe with lung ?", for example, the pattern for "John" is presented to IS-A network through $KN(x)$ and the pattern for "breathing with lung" is presented to Property-Value network through $TP(n_1)$, and the superimposed pattern of "John", "person", and "mammal" will be retrieved from IS-A network in the same way as described above. Although multiple facts are associated with these concepts, only the pattern for "breathing with lung", which is being presented through $TP(m)$ shown in Fig. 3.3, will be retrieved from Property-Value network by M["mammal"]="breathing with lung", since the activation of the other facts are completely supressed by the inhibitory element I_{KV}. The fact that a pattern can be retrieved from Property-Value network means

that the answer for this query is "yes". Oppositely, in the case that a query is not true or no knowledge that can be used to answer a query is stored in the model, no pattern can be retrieved from Property-Value network. It means that the answer is "no".

However, when a property of a concept has multiple values, the model cannot know how many values the property of the concept has and cannot distinguish whether the retrieved value is a ture answer. For example, from the inheritance hierarchy shown in Fig. 3.2, we know that the color of person has three values: white, yellow, and black. If we introduce readout control units (RCUs) in Property-Value network (Hirai, 1983), for the query, "What is the color of person", the present model can retrieve the three values one by one. However, for the query about the color of Bill, since the pattern for"person" will also appear at the output of IS-A network by M["Bill"]="person" and be fed to Property-Value network, after value "pink" is retrieved by M["Bill","color"]="pink", the other three values: "white", "yellow", and "black", will also be retrieved by activating RCU. Evidently, these values are not the correct answer as the color of Bill.

In order to solve this problem, an alternative to the present model can be considered. Structure of the alternative scheme is shown Fig. 3.4. To answer some property of a concept, the concept can be retrieved from IS-A network and be fed to Property-Value network. If the fed concept is not the one directly associated with the property, no pattern will be retrieved from the Property-Value network, and it's superordinate concept will be further retrieved from IS-A network and be fed to Property-Value network. Such process will be repeated until the concept

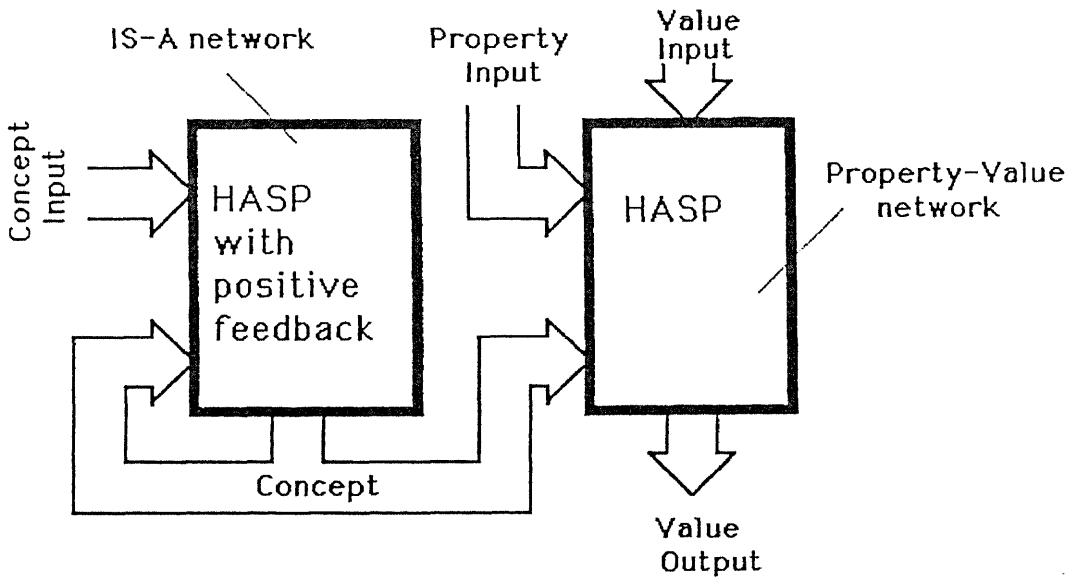


Fig. 3.4. Structure of an alternative model for query-answering. Network notations are the same as those in Fig. 2.1.

that is directly associated with the property is retrieved. Hence, by the finally retrieved concept and the property , the values directly associated with them will be retrieved one by one correctly. However, this scheme takes very long time to access the concept that is directly associated with the property and the advantages of parallel structure can not be taken sufficiently.

3.3.2 A network for recognition

Recognition problem is to locate the concept that completely or most closely exhibits all of the observed features in the stored knowledge base. Suppose, for example, that we have the knowledge shown in Fig. 3.1 and that we are seeing something having four legs, long nose, big ears and so on, we can recognize it as an African elephant immediately. This section describes a network that can perform such recognition tasks.

Structure of the network is shown in Fig. 3.5. It consists of four associative networks: Property-Concept network, Reverse IS-A network, working memory, and IS-A network. Property-Concept network and Reverse IS-A network are constructed by a heteroassociative network of HASP with positive feedback. The working memory and IS-A network are constructed by a HASP, which has positive feedback loops but has not readout control units. Next, we describe how the model memorizes the inheritance hierarchy with exceptions shown in Fig. 3.2 and how it searches the concepts in the hierarchy to answer recognition problems. In Property-Concept network, all explicit associative relationships from property to concept or the associative relationships from fact to concept shown in Fig. 3.2 are stored in $W_{SK}(y_2, x)$: M["nutritive source, organic matter"]="mammal", M["standing

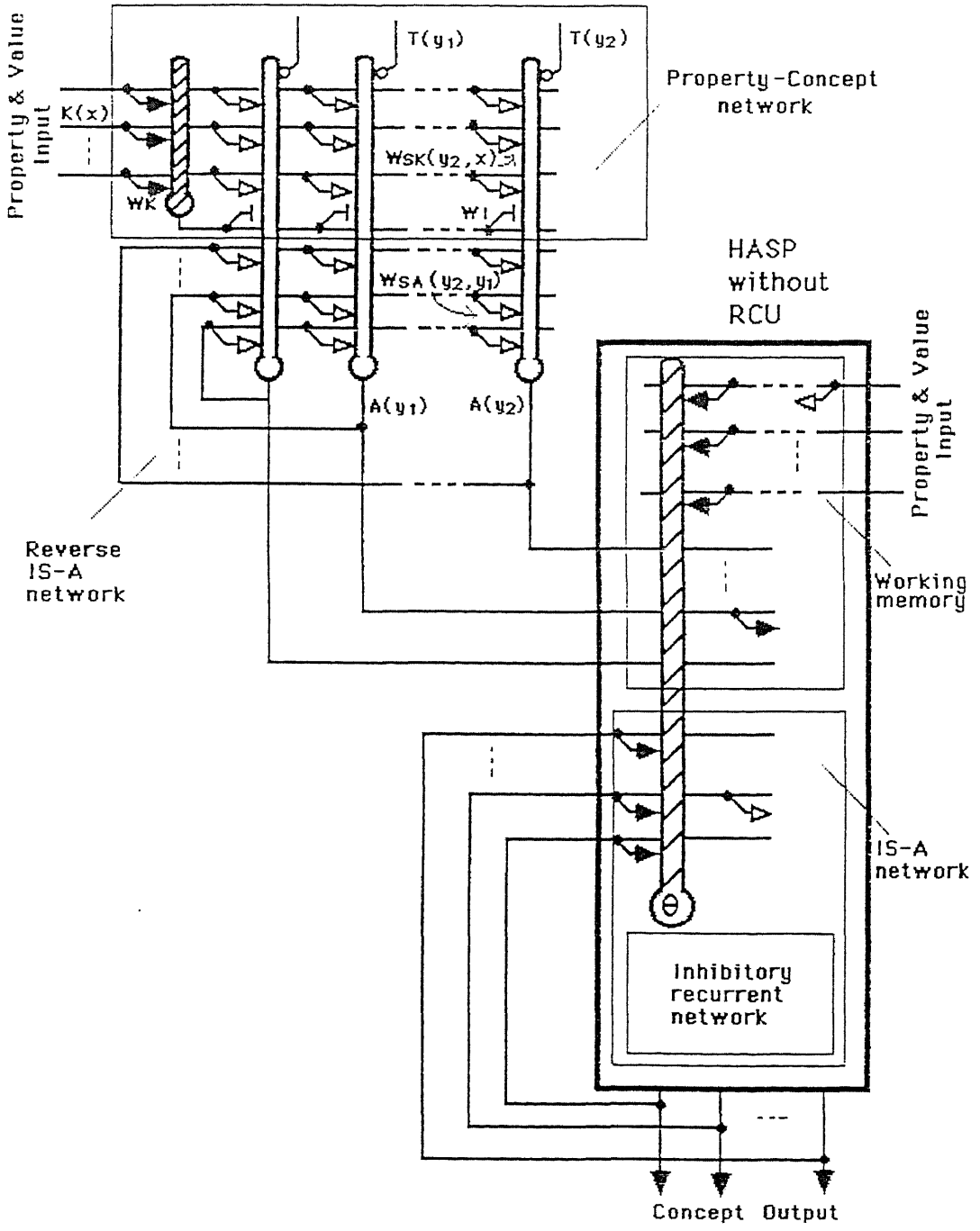


Fig. 3.5. Structure of the network for recognition. Network notations are the same as those in Fig. 2.1.

erect"]="person", and so on. In Reverse IS-A network, all associative relationships from the superordinate concepts to the subordinate ones are stored in $W_{SA}(y_2, y_1)$: M["mammal"]="person", M["person"]="John", and so on. In IS-A network, all associative relationships from the subordinate concepts to the superordinate ones are stored in the modifiable excitatory connections of the network: M["John"]= "person", M["person"]="mammal", and so on and the autocorrelations of patterns for the all concepts are established in the modifiable inhibitory connections of the network. A recognition task: Recognize a concept that has following features: "It's nutritive source is organic matter" and " It stands errect" from the hierarchy shown in Fig. 3.2, for example, can be performed as follows. By presenting the first feature "It's nutritive source is organic matter", to Property-Concept network through $K(x)$, concept "mammal" can be retrieved by M["nutritive source,organic matter"]="mammal" first. Concept "mammal" will further activate it's subordinate ones, "person" and "elephant", and the "person" will activate it's subordinate ones, "John" and "Bill", through the positive feedback loops, since the all associtive relationships from the superordinate concepts to the subordinate ones are stored in $W_{SA}(y_2, y_1)$ of Reverse IS-A network. Finally, a superimposed pattern of all concepts, "mammal", ..., "Bill", will appear at the output of Reverse IS-A network. The elements of Reverse IS-A network are connected with those of the working memory by fixed excitatory connections and these connections are chosen so that the active elements of Reverse IS-A network can activate the corresponding elements of the working memory. Therefore, the superimposed pattern appearing at the output of Reverse IS-A network can activate the corresponding

elements of the working memory and can be memorized in the working memory by using the first feature as a key pattern according to the learning algorithm $L-2.6$ of working memory as described in Section 2.5. After that, all elements of the networks are reset. Next, by presenting the second feature, "It stands erect", to Property-Concept network through K(x), a superimposed pattern of "person", "John", and "Bill" will appear at the output of Reverse IS-A network in the same way as described above and will be stored in the working memory by using the second feature as a key pattern. And then, by presenting the superimposed pattern of the two features to the working memory, the superimposed pattern of "person", "John", and "Bill" will appear at the output of the working memory by the intersection operation of the two sets of concepts: M["nutritive source, organic matter", "standing erect"]=M["nutritive source, organic matter"]$\cap M$["standing erect"]={"mammal", "person", "elephant", "John", "Bill"}$\cap${"person", "John", "Bill"}={"person", "John", "Bill"} (Hirai, 1983). Since the associative relationships from the subordinate concepts to the superordinate ones: M["Bill"]="person" and M["John"]="person" are stored in IS-A network, the concept "person" , which is the uppermost one among the three concepts in the IS-A hierarchy, will win the competition through the inhibitory recurrent network and appear at the output of IS-A network finally. Therefore, the concept having features, "It's nutritive source is organic matter" and "It stands erect", can be recognized as "person".

Since recognitions are performed by the intersection operation for the multiple sets of concepts as described above, in the case that no concept in the stored knowledge base can be matched to completely exhibit the *all* observed features,

no concept can be retrieved by the present model as an answer. In fact, however, since the intersection operation is implemented by the inhibitory element of HASP (Hirai, 1983), by providing adequate inhibition to the inhibitory element of the working memory, the output of the inhibitory element is reduced, and the concept that most closely match the observed features can be retrieved. That is, the model can perform the "best match" operation.

It should be noted that the model is assumed to be controlled by a control network, from which the commands such as "Present the first feature to the model", "Present the superimposed key pattern to the working memory", or "Reset the elements of the networks" are sent out. Such a control network can be constructed by a procedural memory as described in Section 2.4 and is assumed to exist in the model.

3.3.3 Comparisons between our model and the previous ones

In Hinton's model (1982), by choosing distributed representations appropriately, property inheritance can be spontaneously established without any specific connections. That is, by representing a superordinate concept with a pattern of microfeatures (individual active units) that is simply the set of microfeatures common to the patterns that represent it's subordinate ones, any effects caused by the pattern for the superordinate concept will automatically transfer to the patterns for it's subordinate ones. Therefore, by encoding elephant as a pattern (111000 000000) and Clyde as (111000 111000), for example, the hierarchical relation between them is established implicitly and the properties that elephant have can be inherited to

Clyde naturally. Evidently, such an encoding work is artificial and troublesome. In our model, the property inheritance is ensured by establishing associative relationships from the subordinate concepts to the superordinate ones without necessity to distinguish between the superordinate concepts and the subordinate ones to encode them as Hinton's model adopted.

Moreover, in recognition problem, if more than one concept that possesed the all observed features are found, Fahlman's NETL assumes that the one that is uppermost in the IS-A hierarchy is accepted. However, NETL itself does not have a function to distinguish which concept is the uppermost one in the IS-A hierarchy and to select it as an answer. In our model, as described in Section 3.3.2, when multiple candidates (e.g. "person", "John", and "Bill") for a recognition problem appear at the output of the working memory, the uppermost one, "person", in the IS-A hierarchy can be retrieved from the IS-A network.

Finally, NETL is incapable of supporting the "best match" operation, because the marker passing system was adopted (Brachman, 1985; Fahlman, 1982). However, the present model for recognition can perform such an operation by lowering selection criterion of the model as described in Section 3.3.2.

3.4 Results of Simulation

The inheritance hierarchy with exception shown in Fig. 3.2 was taken as an example and the query-answering and recognition by the model were simulated on a digital computer to see weather the model can perform them correctly and effectively. The patterns representing concepts, "Bill", ..., "mammal", were assumed to be arrays

composed of 5 elements as follows: "Bill"=(1, 0, 0, 0, 0), ..., "mammal"=(0, 0, 0, 0, 1). The patterns representing properties, "color", "occupation", and "nutritive source" were assumed to be arrays composed of 3 elements as follows: "color"=(1, 0, 0), "occupation"=(0, 1, 0), "nutritive source"=(0, 0, 1). The patterns representing values of properties or facts, "doctor", ..., "pink", were assumed to be arrays composed of 8 elements as follows: "docotor"=(1, 0, ..., 0), ..., "pink"=(0, ..., 0, 1).

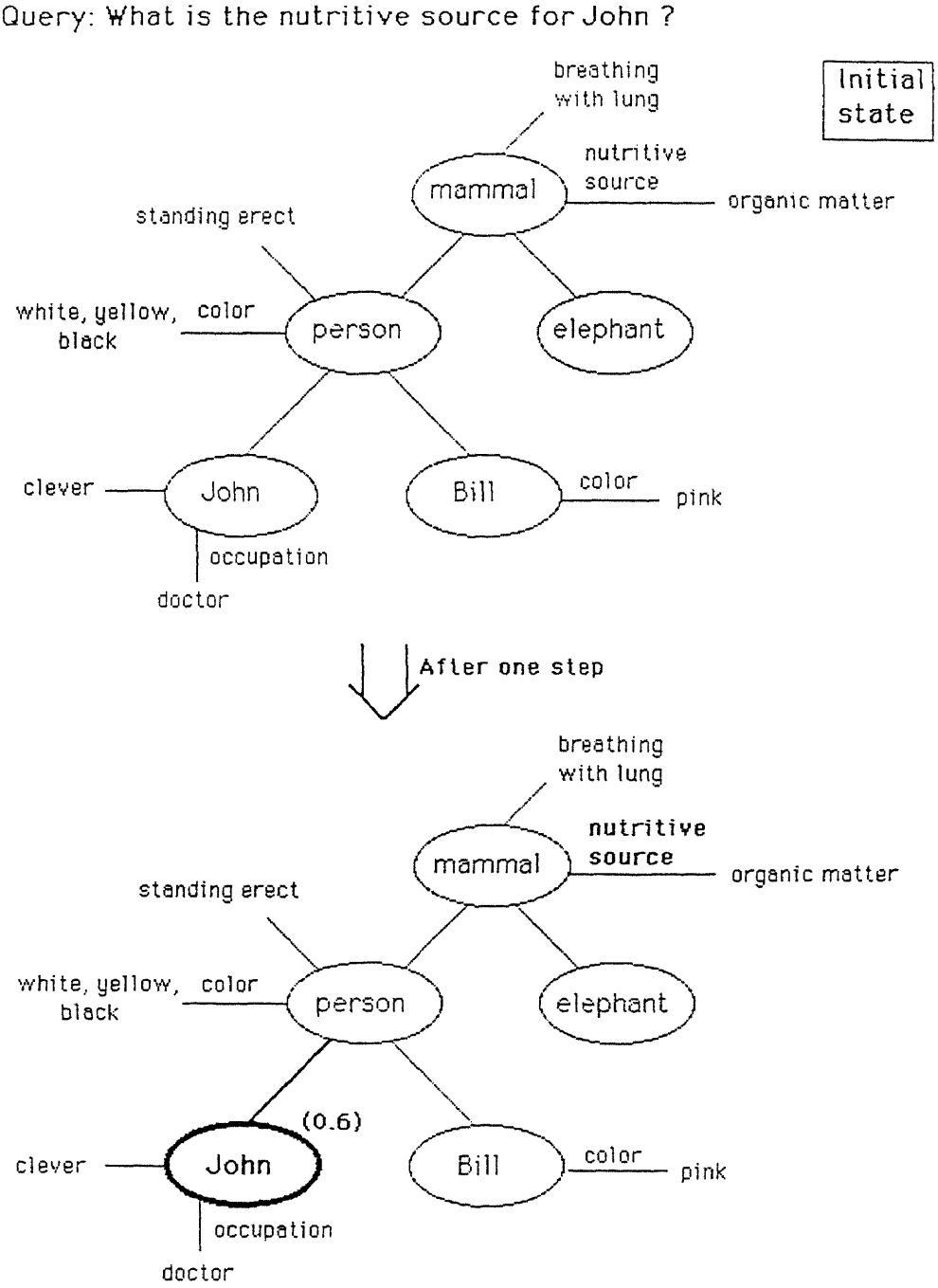
3.4.1 Parameters of the model

Parameters of the model used in the computer simulation are as follows. The time constant of the excitatory elements in all the heteroassociative networks is set to 1.0 time units of numerical calculation. The time constant of the excitatory elements in the inhibitory recurrent networks of the network for query answering and the network for recognition are set to 2.0 and 5.0 time units of numerical calculation respectively. A saturation characteristic is introduced to the excitatory elements in the heteroassociative networks that have positive feedback loops. The saturation value is set to 1.0. The strength of the modifiable excitatory connections in all networks except the working memory are incresed from 0 to 1.0 by learning. The initial strength of the modifiable inhibitory connections in all the inhibitory recurrent networks is set to 1.5 and reduced to 0 by learning. The strength of the fixed excitatory connections from input lines to inhibitory elements (e.g. W_K shown in Fig. 3.3) is set to 1.0. The strength of the fixed inhibitory connections from inhibitory elements to excitatory ones (e.g. W_I shown in Fig. 3.3) is set to 1.0. The threshold, θ, of the inhibitory element in HASP without RCU shown in Fig. 3.5

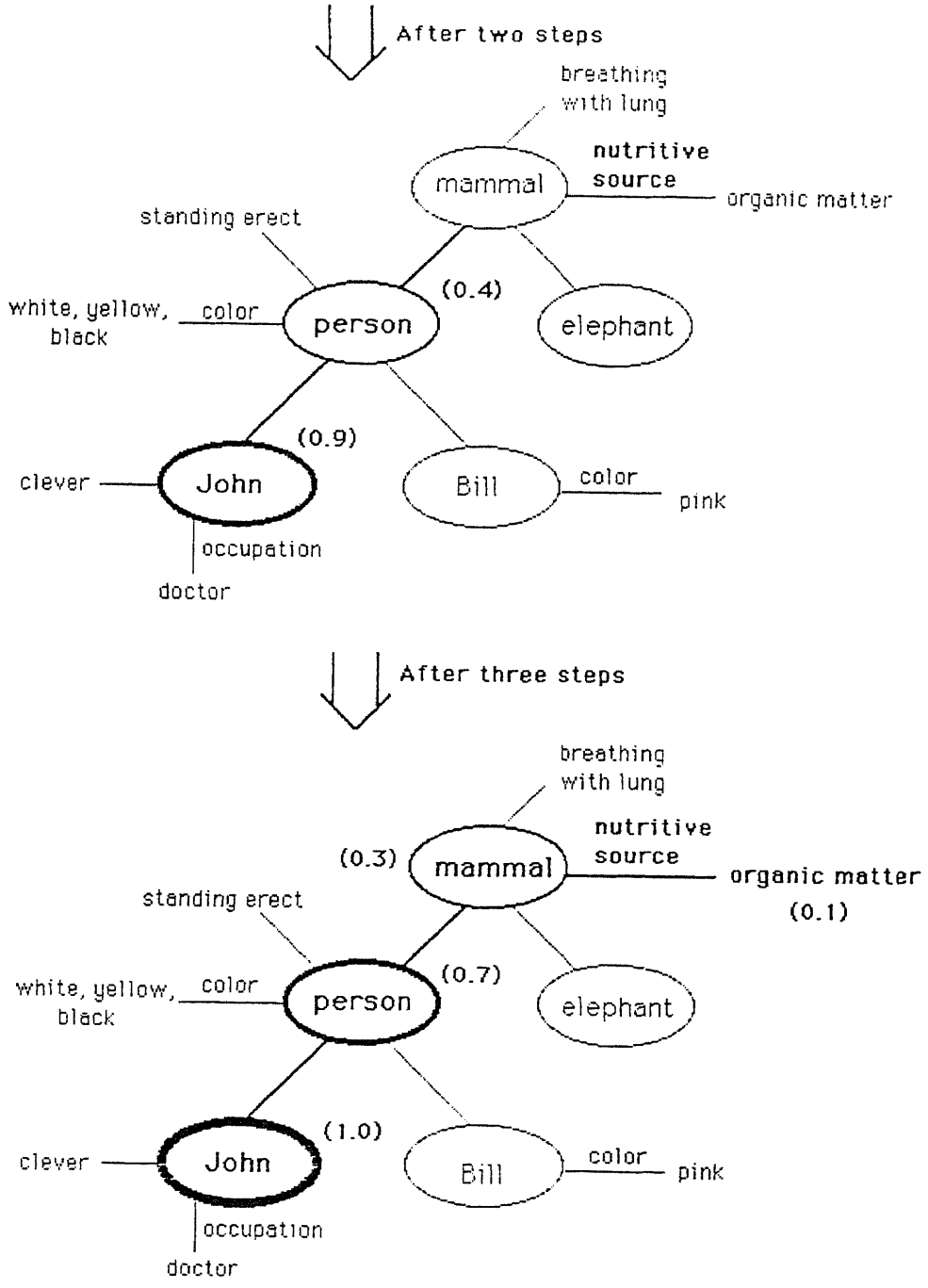
is set to 1.0. The strength of the fixed excitatory connections from the excitatory elements of the heteroassociative networks to the recurrent inhibitory networks is set randomly to a value between 1.0 and 1.1. For the working memory, decaying coefficient b and strengthing coefficient c of the excitatory connections in (2.22) are set to 0.999 and 1.0 respectively. The saturation strength of the connections, W, in (2.23) is set to 1.0.

3.4.2 Results of simulation for query answering

Suppose the model for query answering memorized the inheritance hierarchy with exception shown in Fig. 3.2 as described in Section 3.3.1. A result of simulation for answering a "what" type query, "What is the nutritive source for John ?", is shown in Fig. 3.6. From this figure we know that no concept and value are active in the initial state. That is, no pattern for concepts and values appears at the outputs of IS-A network and Property-Value network of the model. By presenting the pattern for concept "John" and the pattern for property "nutritive source" to the model as described in Section 3.3.1, after one step of numerical calculation, the pattern for "John" is activated and is retrieved from IS-A network with magnitude 0.6. After two steps of numerical calculation, concept "person" is activated with magnitude 0.4. After three steps of numerical calculation, concept "mammal" is activated with magnitude 0.3 and the pattern for the value "organic matter" begins to be activated and to appear at the output of Property-Value network with magnitude 0.1. After 10 steps of numerical calculation, the active magnitude of the value "organic matter" is increased to a magnitude, near by 1.0. Hence, "organic matter" is the answer for the query. Besides, for query "What is color of Bill ?",


Query: What is the nutritive source for John ?
Initial state
breathing with lung
nutritive source
mammal
organic matter
standing erect
white, yellow, black
color
person
elephant
clever
John
Bill
color
pink
occupation
doctor
After one step
breathing with lung
nutritive source
mammal
organic matter
standing erect
white, yellow, black
color
person
elephant
(0.6)
clever
John
Bill
color
pink
occupation
doctor

(continued)


After two steps
breathing with lung
nutritive source
mammal
organic matter
standing erect
(0.4)
white, yellow, black
color
person
elephant
(0.9)
clever
John
Bill
color
pink
occupation
doctor
After three steps
breathing with lung
nutritive source
(0.3)
mammal
organic matter
(0.1)
standing erect
white, yellow, black
color
(0.7)
person
elephant
(1.0)
clever
John
Bill
color
pink
occupation
doctor

(continued)

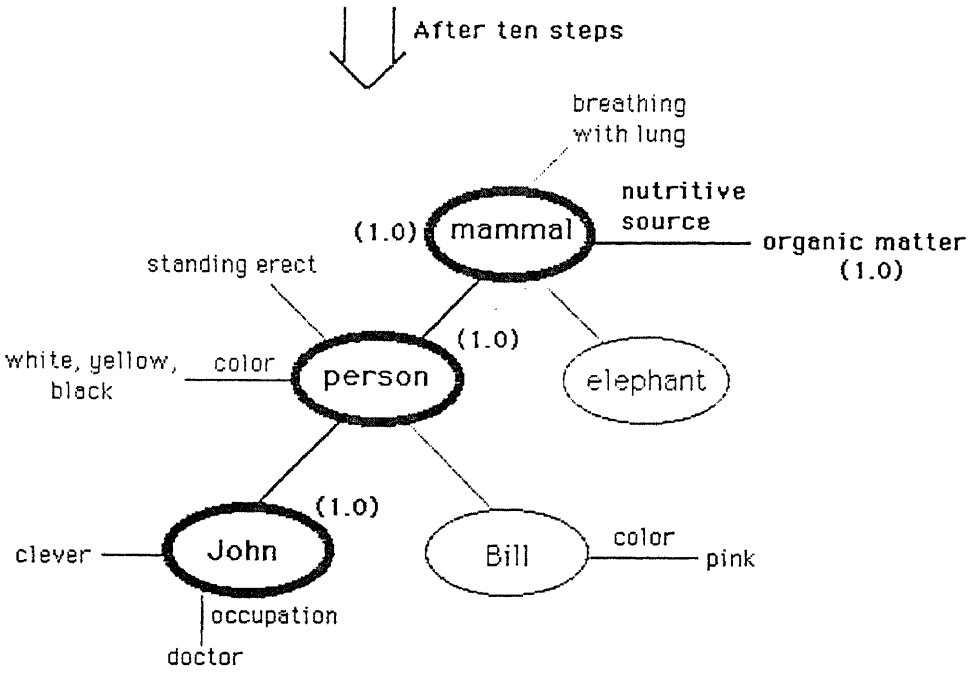


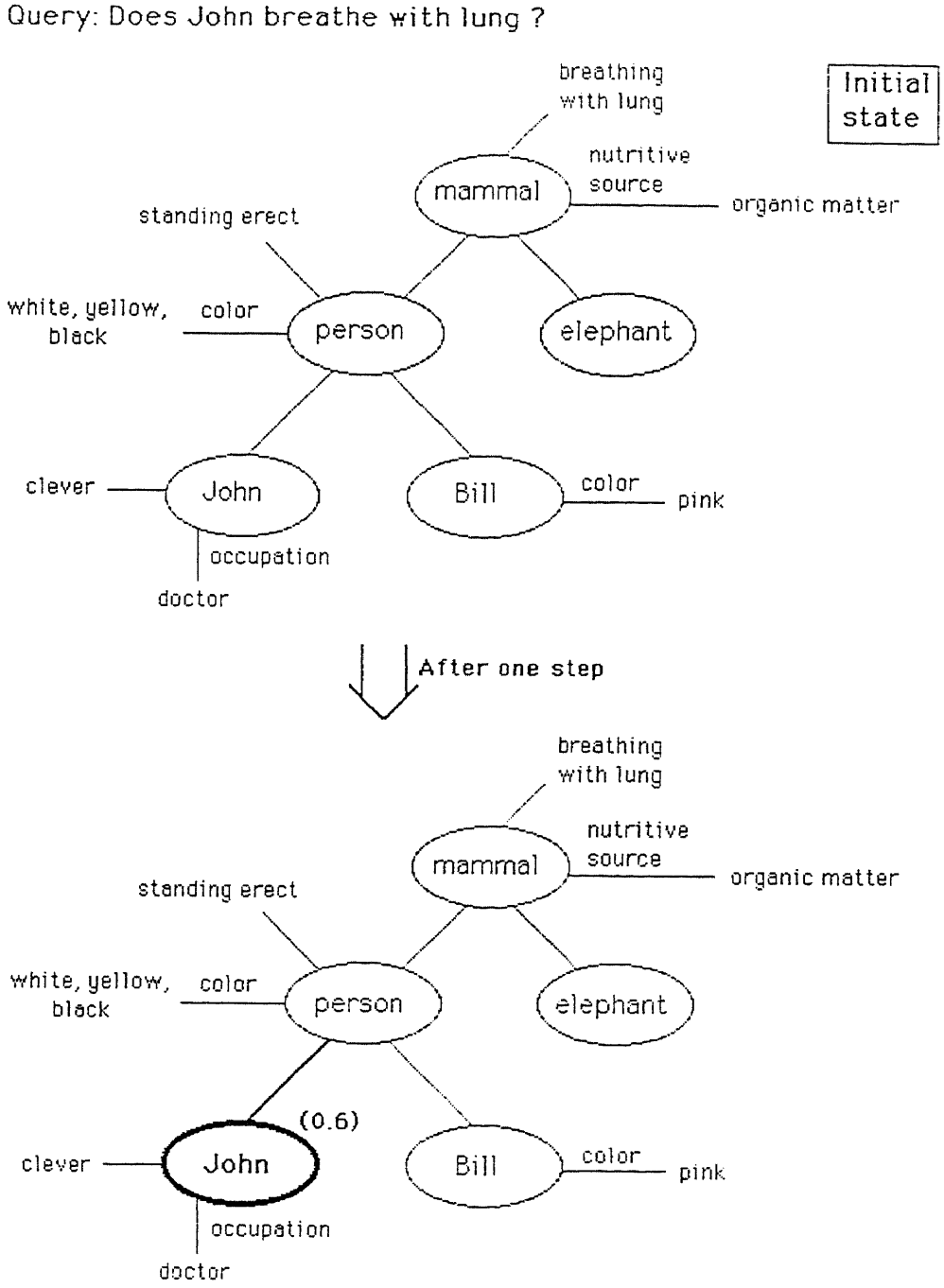
Fig. 3.6. A result of simulation for answering a "what" type query.

although the explicit information "The color of Bill is pink", which is used to answer the query, shown in Fig. 3.2 is an exception to the property inheritance as described in Section 3.3.1, from the result of the simulation we know that answer "pink" can be obtained through eight steps of nemerical calculation. On the other hand, for the queries about the color of person or the color of John, one of the answers: white, yellow, and black can be obtained very effectively without any interferences caused by the exception.

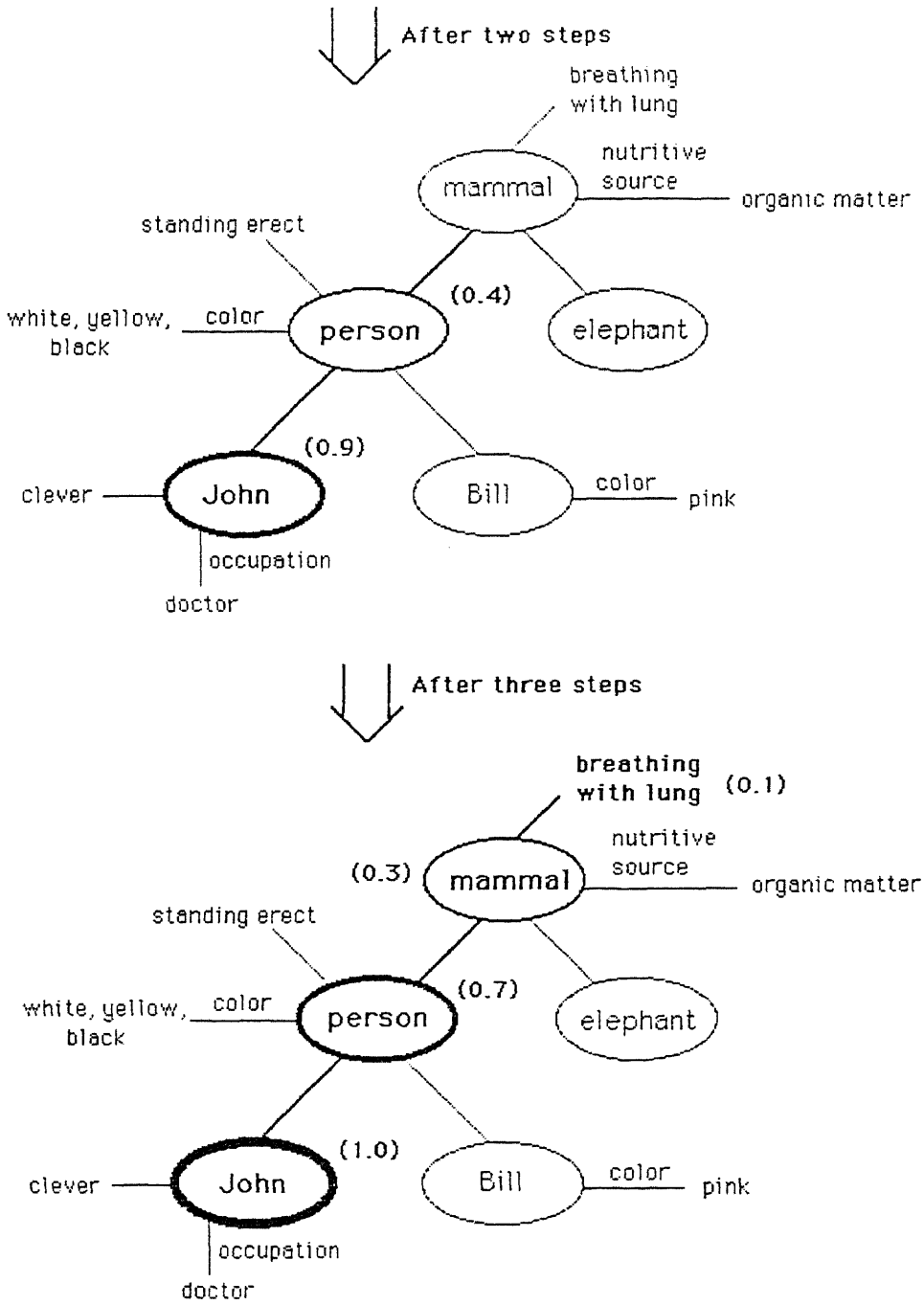
An example of simulation for answering a "yes-no" type query, "Does John breathe with lung ?", is shown in Fig. 3.7. As shown in the figure, by presenting the pattern for concept "John" and the pattern for fact "breathing with lung" to the model, after three steps of numerical calculation, the concepts, "John", "person", and "mammal", are activated in order, and fact "breathing with lung" begins to be activated and to appear at the output of Property-Value network with magnitude 0.1. After 10 steps of numerical calculation, the active magnitude of value "breathing with lung" is increased to a magnitude, near by 1.0. Hence, the answer for the query is "yes". Another example of simulation for answering a "yes-no" type query is shown in Fig. 3.8. The query is "Is Bill clever ?". As shown in the figure, no matter how many steps of numerical calculation are performed, no pattern for any value can be activated. Hence, the answer for the query is "no".

3.4.3 Results of simulation for recognition

A result of simulation for performing a recognition task is shown in Fig. 3.9. The recognition task is to recognize a concept that posses following two features: "It's nutritive source is organic matter" and "It stands erect" from the inheritance


Query: Does John breathe with lung ?
Initial state
breathing with lung
mammal
nutritive source
organic matter
standing erect
white, yellow, black
color
person
elephant
clever
John
Bill
color
pink
occupation
doctor
After one step
breathing with lung
mammal
nutritive source
organic matter
standing erect
white, yellow, black
color
person
elephant
(0.6)
clever
John
Bill
color
pink
occupation
doctor

(continued)


After two steps
breathing with lung
nutritive source
mammal
organic matter
standing erect
(0.4)
white, yellow, black
color
person
elephant
(0.9)
clever
John
Bill
color
pink
occupation
doctor
After three steps
breathing with lung
(0.1)
nutritive source
(0.3)
mammal
organic matter
standing erect
(0.7)
white, yellow, black
color
person
elephant
(1.0)
clever
John
Bill
color
pink
occupation
doctor

(continued)

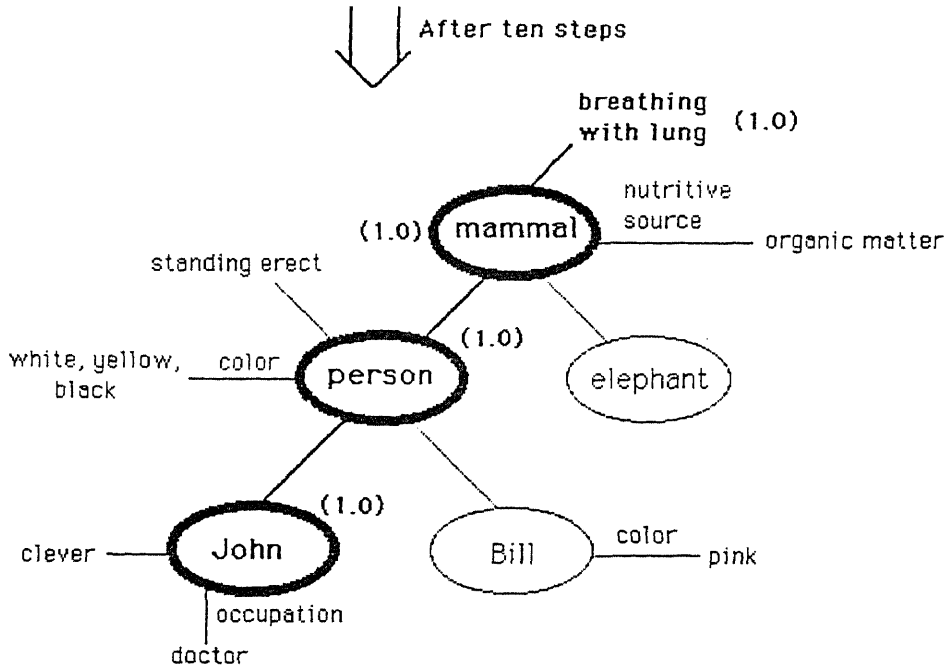


Fig. 3.7. A result of simulation for answering a "yes-no" type query.

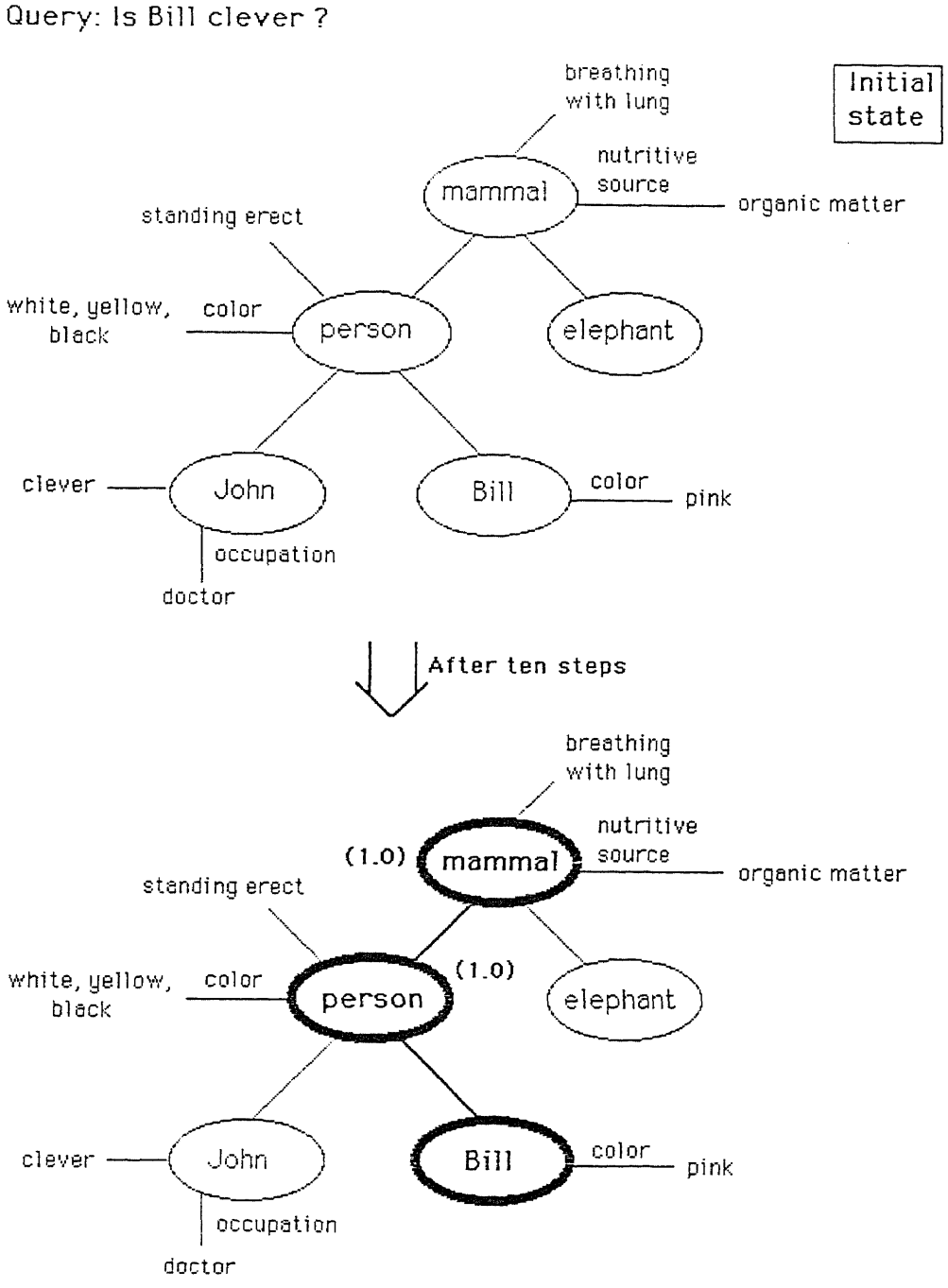
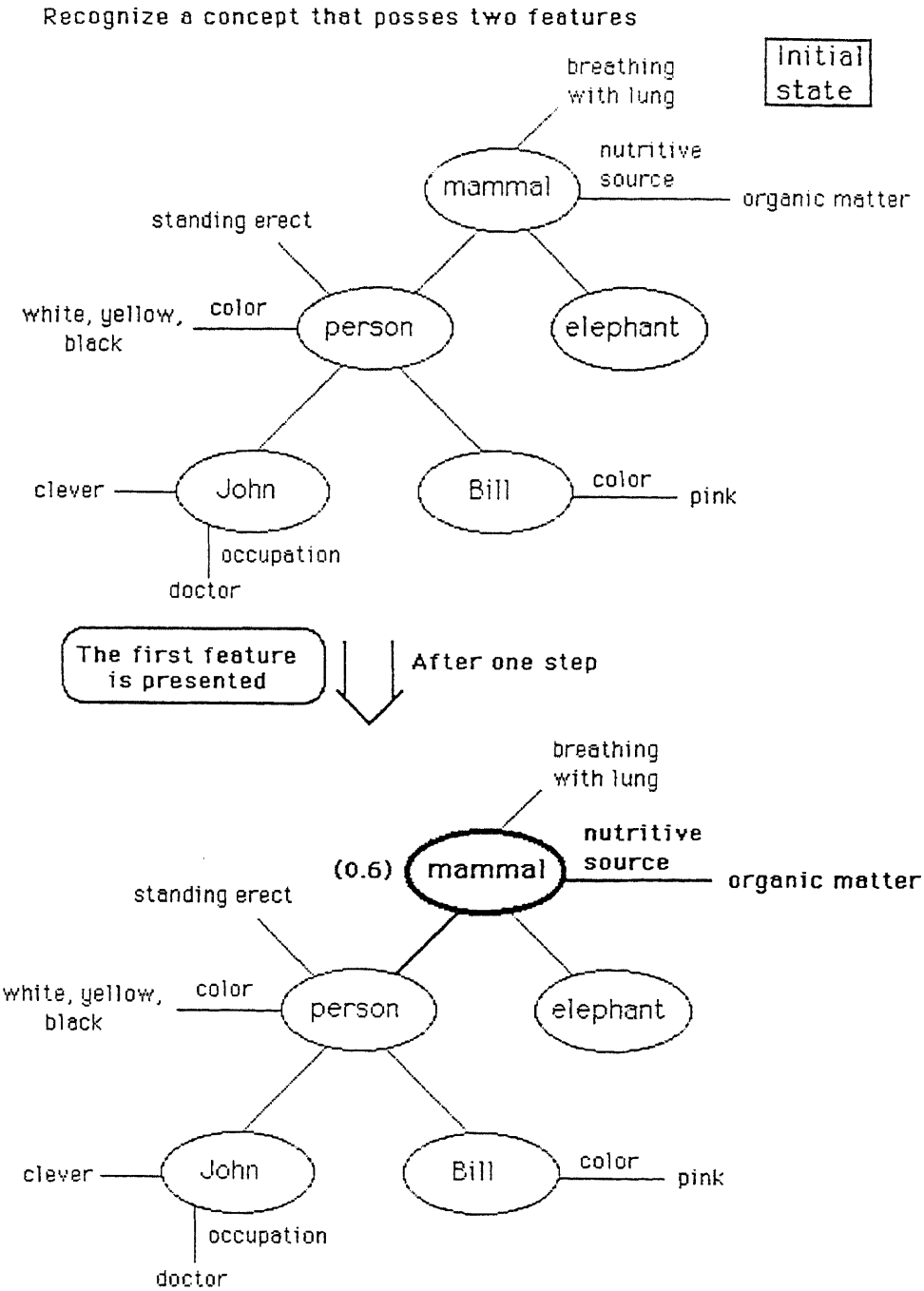
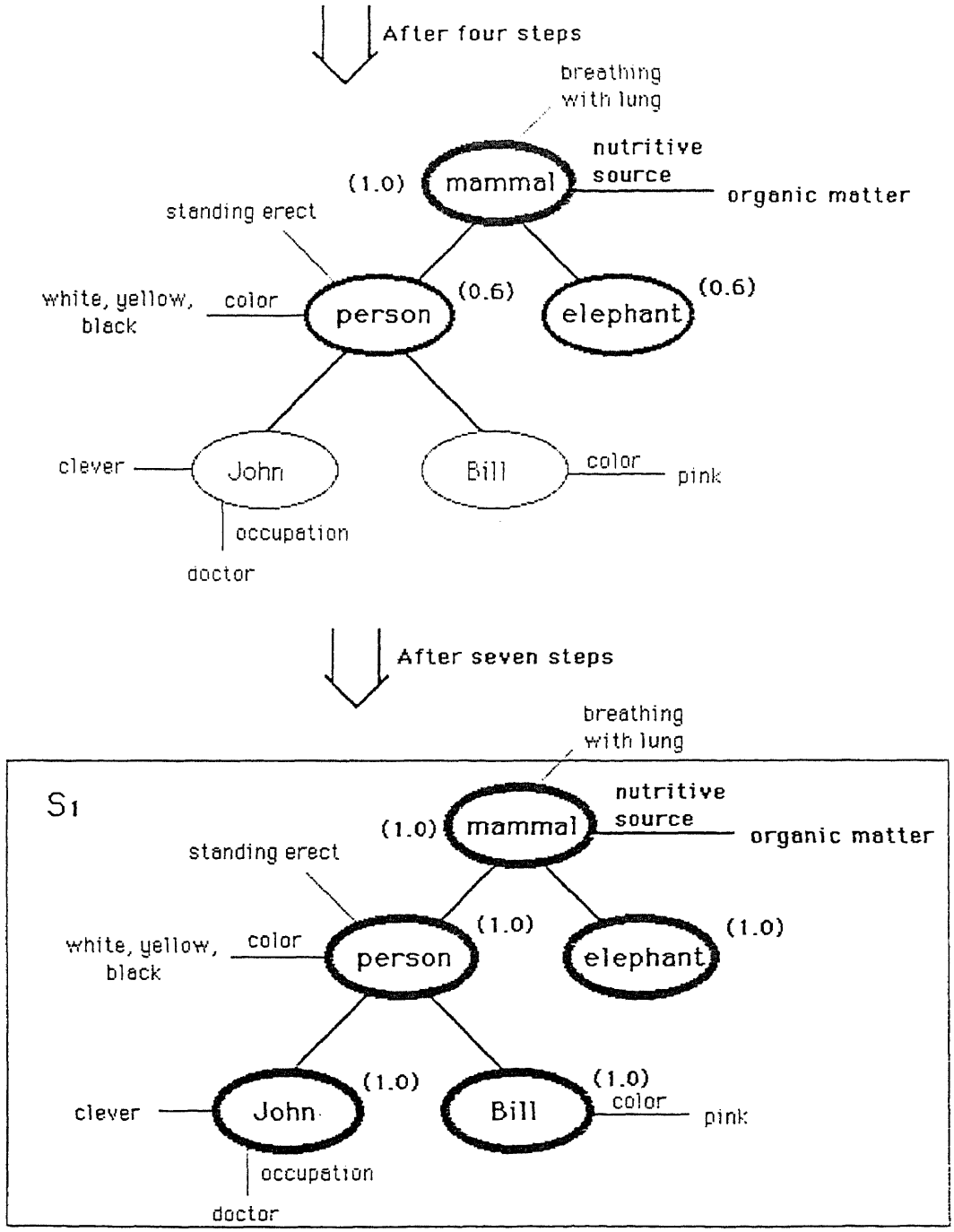


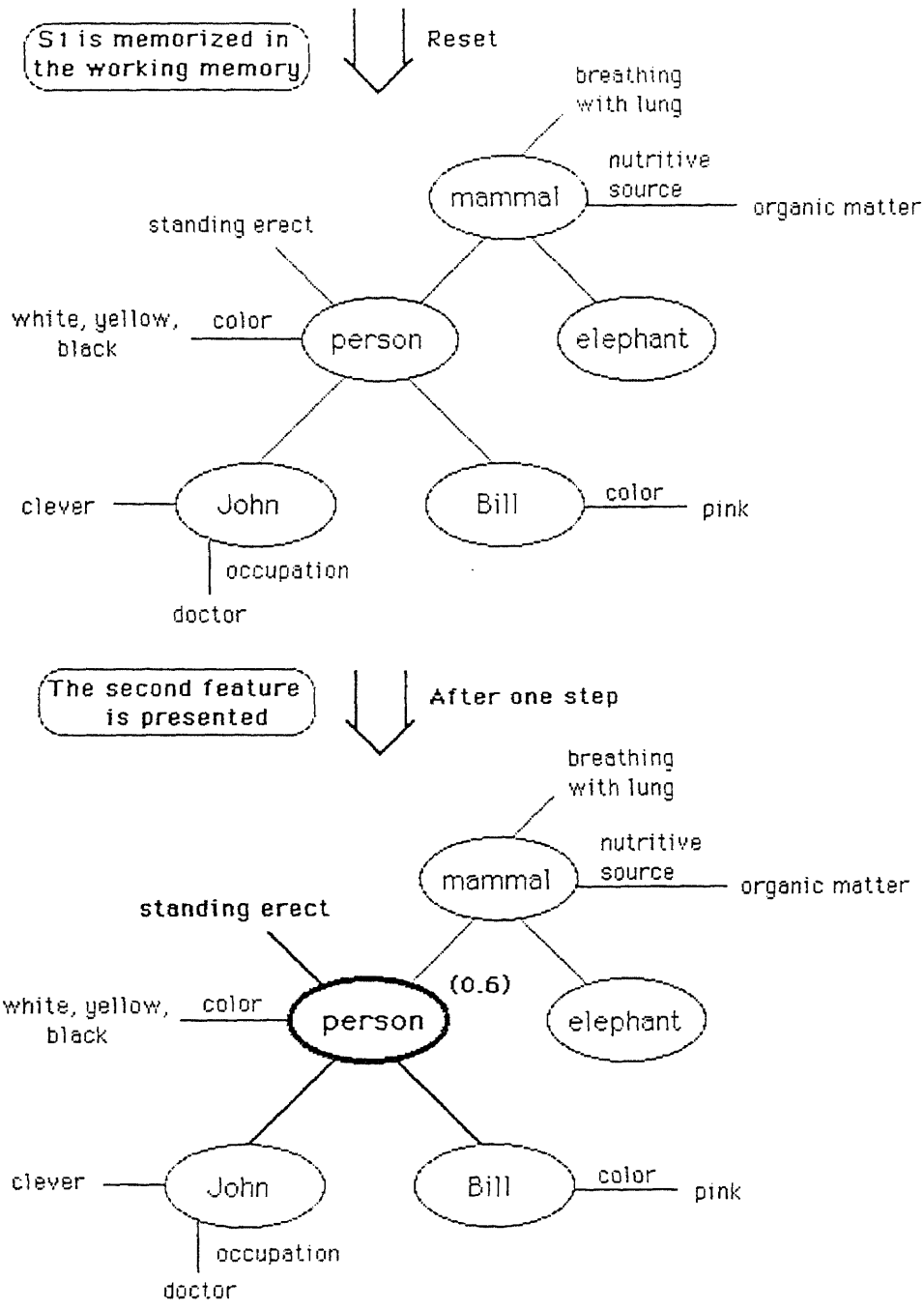
Fig. 3.8. A result of simulation for an unknown query.


Recognize a concept that posses two features
Initial state
breathing with lung
nutritive source
mammal
organic matter
standing erect
white, yellow, black
color
person
elephant
clever
John
Bill
color
pink
occupation
doctor
The first feature is presented
After one step
breathing with lung
(0.6)
mammal
nutritive source
organic matter
standing erect
white, yellow, black
color
person
elephant
clever
John
Bill
color
pink
occupation
doctor

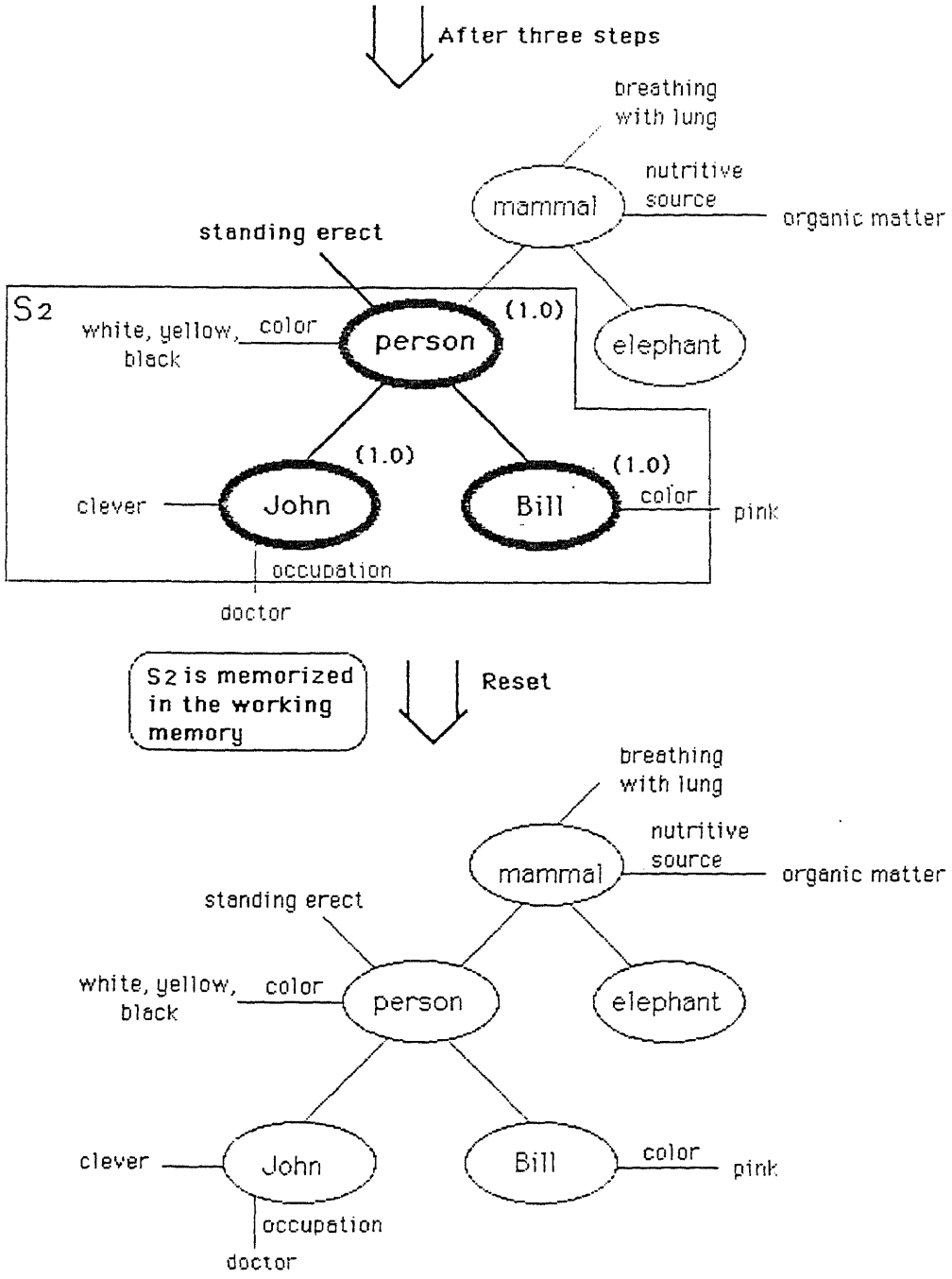
(continued)


After four steps
breathing with lung
nutritive source
(1.0)
mammal
organic matter
standing erect
white, yellow, black
color
person
(0.6)
elephant
(0.6)
clever
John
Bill
color
pink
occupation
doctor
After seven steps
breathing with lung
S1
nutritive source
(1.0)
mammal
organic matter
standing erect
white, yellow, black
color
person
(1.0)
elephant
(1.0)
clever
John
(1.0)
Bill
(1.0)
color
pink
occupation
doctor

(continued)


S1 is memorized in the working memory
Reset
breathing with lung
nutritive source
mammal
organic matter
standing erect
white, yellow, black
color
person
elephant
clever
John
Bill
color
pink
occupation
doctor
The second feature is presented
After one step
breathing with lung
nutritive source
mammal
organic matter
standing erect
(0.6)
white, yellow, black
color
person
elephant
clever
John
Bill
color
pink
occupation
doctor

(continued)


After three steps
breathing with lung
nutritive source
mammal
organic matter
standing erect
S2
white, yellow, black
color
person
(1.0)
elephant
(1.0)
(1.0)
clever
John
Bill
color
pink
occupation
doctor
S2 is memorized in the working memory
Reset
breathing with lung
nutritive source
mammal
organic matter
standing erect
white, yellow, black
color
person
elephant
clever
John
Bill
color
pink
occupation
doctor

(continued)

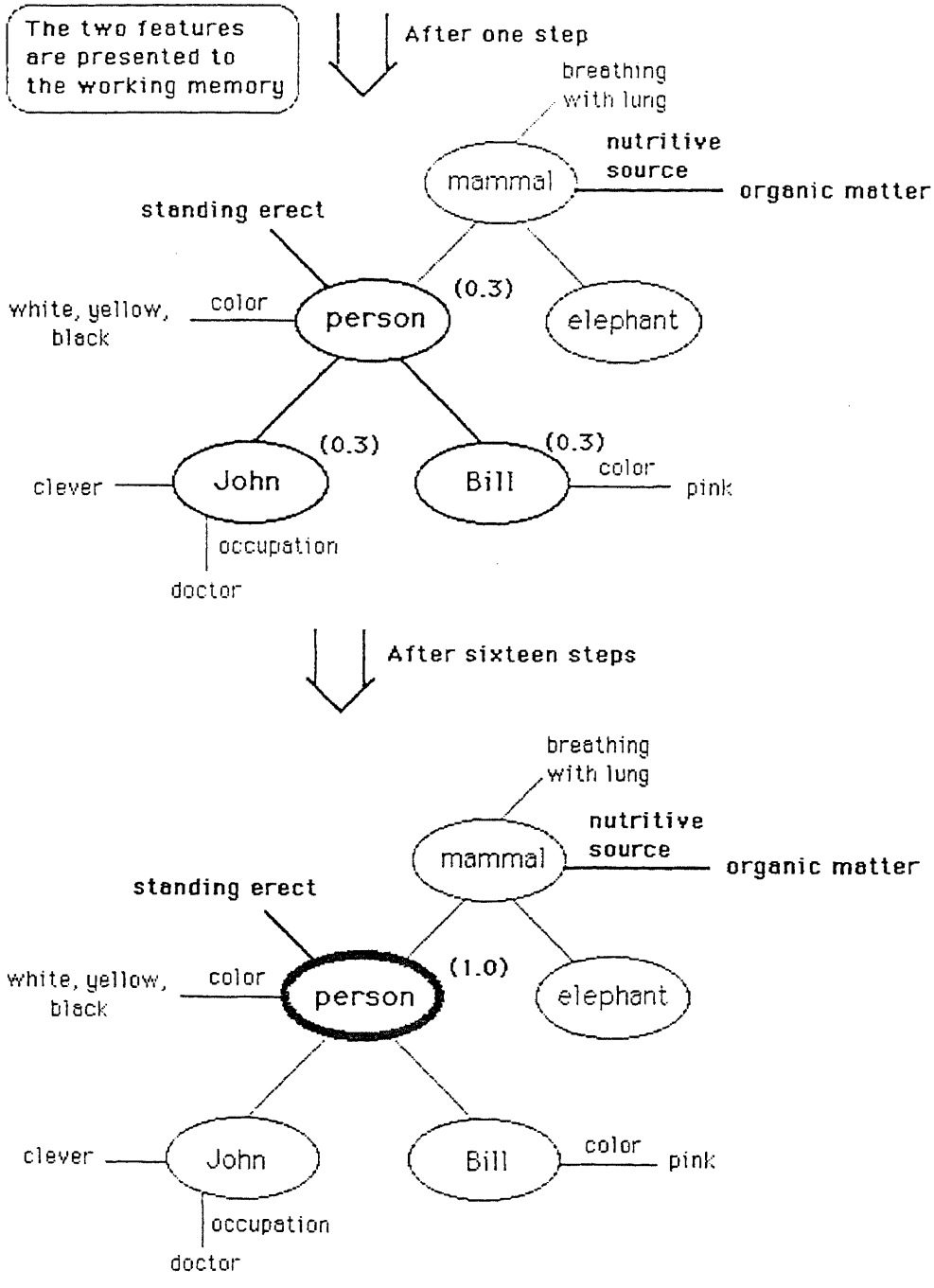


Fig. 3.9. A result of simulation for answering a recognition problem.

hierarchy with exception shown in Fig. 3.2. As shown in the figure, by presenting the pattern for the first feature, "It's nutritive source is organic matter", to Property-Concept network, after one step of numerical calculation, concept "mammal" is activated at the output of Property-Concept network and is memorized in the working memory by using the first feature as a key pattern. After seven steps of numerical calculation, all concepts, "Bill", ..., "mammal", are activated at the output of Property-Concept network. Hence, the set of the five concepts, S_1, is memorized in the working memory by using the first feature as a key pattern. After that, the all elements of the networks are reset. And then, the second feature, "It stands erect" is presented to Property-Concept network. From the figure we know that concept "person" is reactivated after one step of numerical calculation. After three steps, the other two concepts, "John" and "Bill", are reactivated, too. Hence, the set of the three concepts, S_2, is memorized in the working memory by using the second feature as a key pattern. After that, the all elements of the networks are reset again. Next, a superimposed pattern of the two features is presented to the working memory through the key input line. As shown in the figure, the intersection operation for the two set, S_1 and S_2, stored in the working memory is performed and only the common part of the two sets, "person", "John", and "Bill", is activated at the output of the IS-A network with magnitude 0.3 after one step of numerical calculation. After 16 steps, only one concept, "person", remains at the output of the network with magnitude near by 1.0. Hence, "person" is the answer for the recognition problem. In addition, in the case that no concept in the stored knowledge base can be matched to exhibit the all observed features, no

cocept will be activated. For example, for a recognition task: Recognize a concept that has features: "standing erect", "clever", and "The color is pink", no concept can be recognized by the model.

3.5 Summary

In this chapter we have proposed a new scheme for the knowledge representation. By properly connecting associative networks, HASPs, the parallel model that can memorize inheritance hierarchies with exceptions and can be used to answer the two types of queries and recognition problems is constructed. In our models, because the property inheritance is ensured by establishing associative relationships from the subordinate concepts to the superordinate ones, there is neither necessary to directly establish associative relationships from the properties of the superordinate concepts to all the subordinate ones, which will be very parsimonious in implementing the model in true hardware, nor necessary to purposely encode the superordinate concepts with the patterns of microfeatures that are simply the sets of microfeatures common to the patterns that represent their subordinate concepts as Hinton's model adopted. In the case that more than one concept that possesed the all observed features are found in the recognition tasks, the model can select the uppermost one in the IS-A hierarchies (Inheritance hierarchies) as an answer. If no concept that completely posseses the *all* observed features exists, by lowering selection criterion of the model, the model can perform the "best match" operation, and the concept that most closely exhibits all of the observed features can be retrieved as an answer. The computer simulation for the "best match" operation is

left as a future work. The results of simulation show that our model can perform query answering and recognition very quickly and effectively and their performance is not influenced by exceptions to the property inheritance.

Chapter 4

Modeling the Acquisition of Counting

In the acquisition of counting by children, there are three interesting phenomena (Fuson et al., 1982): (1)the number word sequence produced by children can be divided into three distinct portions, called the conventional, stable nonconventional, and unstable portions; (2)irregular number words such as "fifteen" are omitted more often than regular ones such as "fourteen", "sixteen", and "seventeen"; and (3)initially the number word sequence is in a recitation form, rather than in the form of an associative chain of separable serial elements. This chapter at first analyzes these phenomena from the viewpoint of associative memory by assuming the number word sequences are made up of many associative relationships between the number words. This assumption is not contradictory to the third phenomenon described above, because the associative relationships are not confined only to those between the serial number words. On the basis of these analyses, the learning algorithms of the associative network model, HASP, is extended so that it

can mimic some aspects of the learning of sequence which involves the above three phenomena. The learning and production of sequence by the network are simulated on a digital computer, and the results show that the three phenomena can be observed in the performance of the network.

4.1 Motivation

Most of the neural network models proposed so far mimic various human perceptual and cognitive functions such as pattern recognition (eg. Rosenblatt, 1962), associative memory (eg. Hirai, 1983) and even the process of problem solving (Hirai & Ma, 1988). However, it is also necessary to direct our attention to the learning process, through which these functions are acquired. Some work of this type has been carried out. For example, the learning of English past tense, in which three stages can typically be observed, has been simulated by a PDP model (Rumelhart & McClelland, 1986). In the first stage, children learn only a small number of verbs and most of them are irregular. At this stage, children tend to learn the past tense correctly. In the second stage, children learn a much larger number of verbs. These verbs include a few more irregular ones, and the most of them are regular. At this stage, children obtain a linguistic rule of -ed form, but often apply this rule to irregular verbs. In the third stage, the regular and irregular verbs coexist and children regain the correct use of -ed rule.

This chapter describes a neural network model which can mimic some aspects of the learning of sequence such as alphabet sequence, music melody sequence, and the number word sequence. In developmental psychology, a large number of studies

on the development of mathematical thinking have been carried out (Ginsburg, 1983). Among them, several rather interesting types of errors in the acquisition of counting were observed by Fuson et al. (1982). Our model provides an explanation for these psychological phenomena. In order to construct such a model, HASP was modified to simulate general learning. One of the most prominent features of HASP, its ability to represent one-to-many associative relationships, is indispensable for modeling the learning of sequence and for explaining the observed psychological phenomena.

The structure of this chapter is as follows. In Section 4.2, the psychological phenomena occurring in the learning of counting are described. In Section 4.3, these phenomena are analyzed from the viewpoint of associative memory by assuming the number word sequences are made up of many associative relationships between the number words. Section 4.4 describs the HASP with general learning which can mimic the learning of sequence. Section 4.5 presents the results of a computer simulation verifying that the model provides an explanation of the psychological data.

4.2 The Phenomena

Fuson, Richards & Briars (1982) reported several interesting phenomena accompanying the acquisition of counting by children. These are as follows:

[1] The number word sequences up to about thirty produced by children when they are learning to count have a typical structure that is stable across different counting tasks and across several days. The most common form includes an ini-

tial portion that is some beginning part of the conventional sequence (e.g., "one, two, three, four, five"), a middle portion that deviates from the conventional sequence but that is produced with some consistency by a given child (e.g., "seven, nine, ten, twelve"), and a final portion that has little consistency over repeated productions. Table 4.1 shows an example of a typical child's repeated counting trials. In this example, the stable conventional sequence portion is "one two three four" and the stable nonconventional portion is "four six eight nine". The unstable nonconventional portions vary from trial to trial and consist of the words following "nine".

[2] Almost all of the stable nonconventional portions have the words in the conventional orders, but they contain omissions and most of the gaps are of one word. A rather interesting point is that "fifteen" was omitted more often than all other words as shown in Fig. 4.1.

[3] After a stable conventional number word sequence is first acquired it functions as a unidirectional whole structure. The number words can be produced only by reciting the whole sequence. The elaboration of the sequence is a process of differentiation of the words and construction of relations among them. In what they called a recitation context study, Fuson et al. gave 3 and 4 year olds either one, two, or three successive number words as prompts for the production of the next number word. Children who first acquired a stable conventional number word sequence were much more successful with the two word prompt than the one word prompt as shown in Table 4.2, providing evidence that they had not yet elaborated the sequence.

Table 4.1. Example of repeated counting trials of a typical child
(after Fuson et al., 1982)

one	two	three	four	six	eight	nine	fourteen	sixteen	thirteen	five
one	two	three	four	six	eight	nine	twelve	fifteen	sixteen	thirteen
one	two	three	four	six	eight	nine	fourteen			
one	two	three	four	six	seven	eight	nine	eleven		
one	two	three	four	six	eight	nine	fifteen	thirteen	eleventeen	
one	two	three	four	six	eight	nine	sixteen	eight	four	twelve
one	two	three	four	six	eight	nine	thirteen	two	six	
one	two	three	four	six	eight	nine	ten	thirteen	sixty	

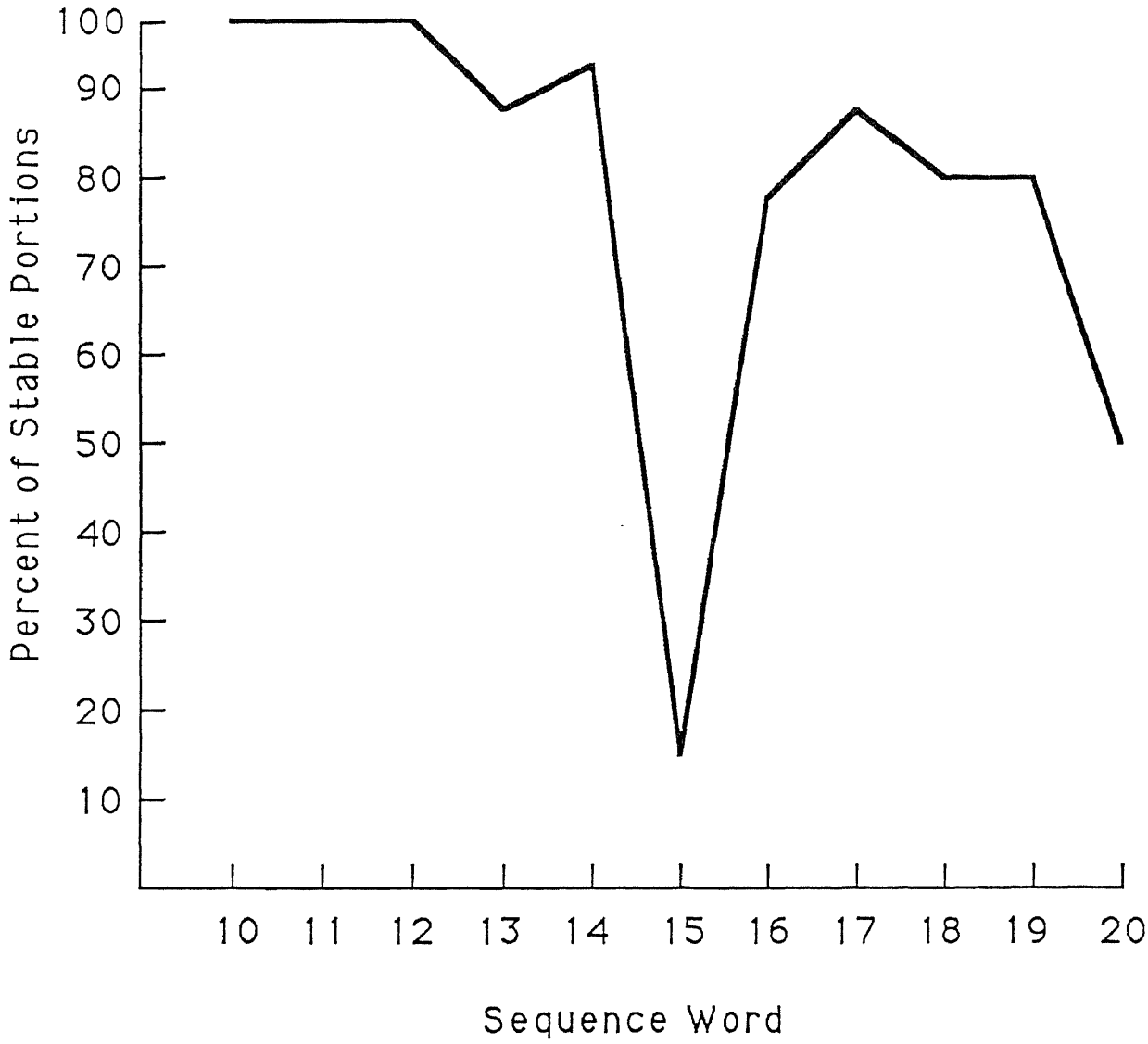


Fig. 4.1. Percentage of conventional and stable portions containing words between ten and twenty produced. This figure includes words from the conventional portion preceding a stable portion (e.g., for 1, 2, ..., 10, 11, 12, 13, 16, 18, 19, from 1 to 13 is conventional and 13, 16, 18, 19 is stable) (after Fuson et al., 1982)

Table 4.2. Percentage of correct responses of children on a recitation contex study (after Fuson et al., 1982)

	One-word stimulus			Two-word stimulus			Three-word stimulus		
Age	Digit	Teens	Mean	Digit	Teens	Mean	Digit	Teens	Mean
3-years	39	15	27	62	48	55	63	33	48
4-years	64	63	63	82	83	82	83	81	82

4.3 Analyses of the Phenomena

That there are three different portions in the number word sequence produced by children can be assumed to reflect the fact that a number word sequence is usually divided into several portions to be learned and exposure to initial portions is more frequent than exposure to later portions, in the early stages of learning.

In this section, we analyze the phenomena described in Section 4.2 from the viewpoint of associative memory.

The learning of a number word sequence such as "1, 2, 3" can be seen as the formation of associative relationships between "1" and "2" and between "2" and "3". Here we introduce an associative mapping function $M[*]$ such that the above two associative relationships can be expressed as follows: $M[\text{"1"}] = \text{"2"}$ and $M[\text{"2"}] = \text{"3"}$. However, as described in Section 4.2, in the production of sequence, the case that "3" is produced next to "1" can be observed. This means that besides the above two associative relationships, $M[\text{"1"}] = \text{"3"}$ must also exist. From this fact we made the following assumption:

> **In the early stages of learning, the remote associative relationship $M[\text{"1"}] = \text{"3"}$ exists and its strength is almost as strong as that of the close one $M[\text{"1"}] = \text{"2"}$. Hence, there is a case that $M[\text{"1"}] = \text{"3"}$ succeeds over $M[\text{"1"}] = \text{"2"}$ in the competition, and "3" is produced immediately after "1". As learning proceeds, the strength of $M[\text{"1"}] = \text{"2"}$ becomes much stronger than that of $M[\text{"1"}] = \text{"3"}$. Hence, "2" is always produced immediately after "1"**

That the number word "fifteen" was omitted more often than all other words may result from its irregular construction (i.e., "fifteen" rather than "fiveteen"). The difficulty of memorizing "fifteen" can be analyzed as follows. From the assumption of remote associative relationship described above it follows that, in the early stages of learning the sequence "fourteen, fifteen, sixteen", the strength of the associative relationship M["fourteen"] = "fifteen" is as strong as that of M["fourteen"] = "sixteen". However, at this point, if the sequence "four, five, six" has been learned, the residual associative relationship M["four"] = "six" will strengthen M["fourteen"] = "sixteen", because "four" and "fourteen" , and "six" and "sixteen" have common parts "four" and "six" respectively, but M["four"] = "five" will not affect M["fourteen"] = "fifteen", because "five" and "fifteen" have no common part. Therefore, the strength of the associative relationship M["fourteen"] = "sixteen" is stronger than that of M["fourteen"] = "fifteen", and "fifteen" is omitted in production of the sequence. As learning proceeds, the strength of M["fourteen"] = "fifteen" becomes much stronger than that of M["fourteen"] = "sixteen", and "fifteen" can be produced normally.

That the production of a number word is much easier when prompted by the preceding two words than when prompted by only the single preceding word can be considered as follows. In the early stages of learning the sequence "1, 2, 3", for example, because both the associative relationships M["1"] = "3" and M["2"] = "3" exist and the strength of the two relationships is almost the same according to the assumption of remote associative relationship, to retrieve "3" by a two-word stimulus "1" + "2", may be much easier than by only a one-word stimulus "2".

However, after learning the sequence "1, 2, 3" many times, the strength of the associative relationship M["1"] = "3" becomes so weak, compared with that of M["2"] = "3", that it can be ignored, and the retrieval of "3" by only "2" may become as easy as by "1"+"2".

4.4 HASP Capable of General Learning

In this section, the HASP which can mimic the learning and production of sequences is described. The learning algorithms of HASP are modified. Structure of the network is shown in Fig. 4.2. It consists of three components: a heteroassociative network, an inhibitory recurrent network and a readout control unit. Each number word is expressed by a spatial pattern which is provided through input lines denoted by $T(y_1)$. Number word sequence is stored in the heteroassociative network and the retrieved number word appears at the output elements denoted by $A(y_1)$. The componments are described below.

4.4.1 Heteroassociative network

While the pattern for a number word is being presented through $T(y_1)$, the network elements $S(y_1)$ and $A(y_1)$ respond according to the dynamic equations to be described later. The response of elements $A(y_1)$, which is forming the present number word at this time, is fed to the network through the positive feedback loops, and the modifiable excitatory connections $W_{SA}(y_1, y_2)$ are strengthened according to the learning algorithm to be described later to store the associative relationship of this number word with itself. And then, if the pattern for the subsequent number word is presented through $T(y_1)$, the serial associative relationship from the

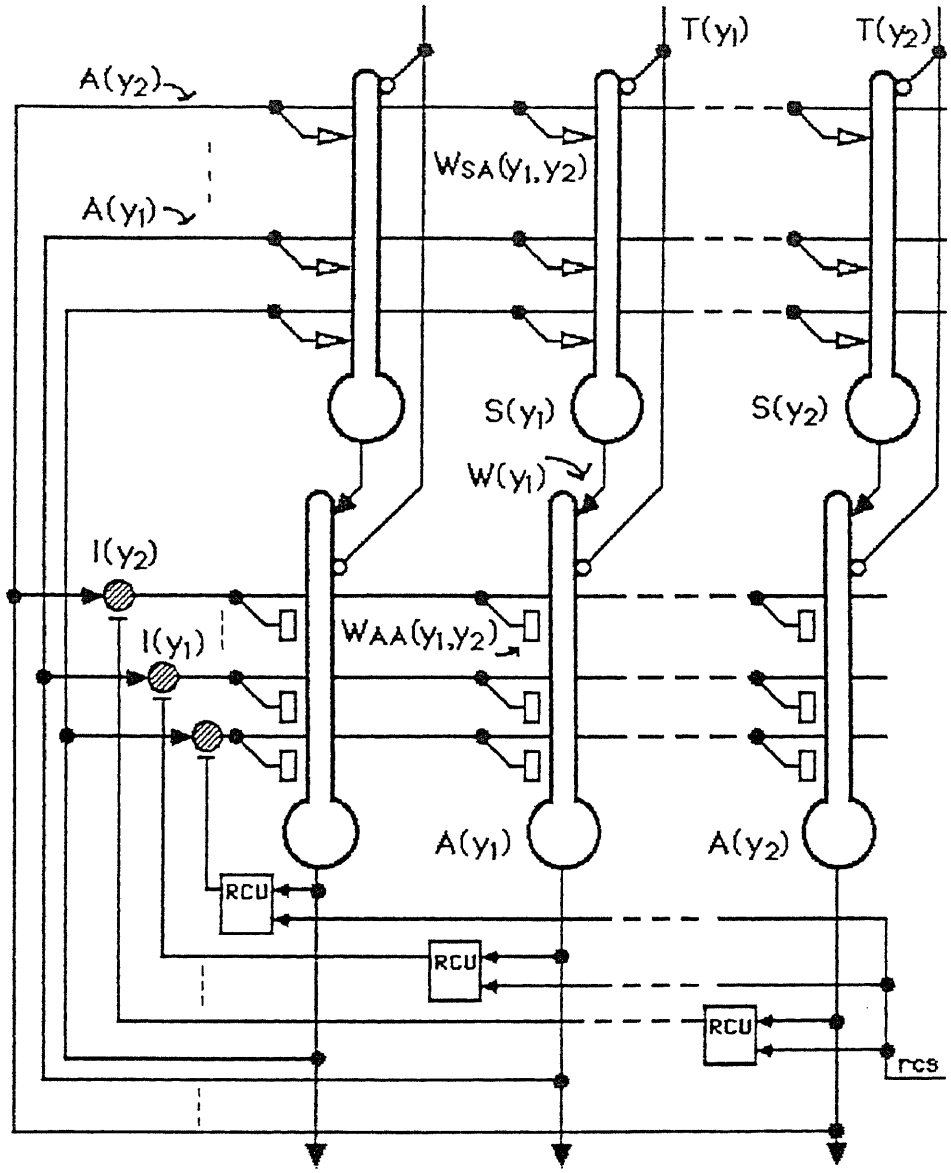


Fig. 4.2. Structure of the modified HASP. Network notations are follows. —□ : modifiable inhibitory connection whose strength is increased by learning. RCU: readout control unit. rcs: readout control signal. The other network notations are the same as those in Fig. 2.1.

preceding number word, which is being maintained by the associative relationship with itself, to the subsequent one are stored in these connections. These connections are gradually strengthened from initial strength, which is assumed to be zero, according to the following learning algorithm (L-4.1):

$$W_{SA}(y_1, y_2, t + \Delta t) = W_{SA}(y_1, y_2, t) + \Delta W_{SA}(y_1, y_2, t)$$

where $\Delta W_{SA}(y_1, y_2, t)$ is the strengthening-rate of connection $W_{SA}(y_1, y_2)$ at time t and is determined as follows:

1. if $A(y_2, t) \cdot T(y_1, t) > 0$ and $W_{SA}(y_1, y_2, t) < WS_1$, then $\Delta W_{SA}(y_1, y_2, t) = V_{WS_1}$. Where WS_1 and V_{WS_1} are two different positive values.

2. if $A(y_2, t) \cdot T(y_1, t) > 0$ and $WS_1 < W_{SA}(y_1, y_2, t) < WS_2$, then $\Delta W_{SA}(y_1, y_2, t) = V_{WS_2}$. Where V_{WS_2} is some positive value less than V_{WS_1} and WS_2 is the saturation value of $W_{SA}(y_1, y_2)$.

3. otherwise, $\Delta W_{SA}(y_1, y_2, t) = 0$.

In addition, $W_{SA}(y_1, y_2)$ decays with time until it reaches its saturation value WS_2.

In this learning algorithm, it is noticeable that the modifiable connections are not strengthened to the saturation value immediately through one shot learning, but are instead strengthened gradually and the strengthening-rate depends on their current strength as shown in Fig. 4.3(a). The reason why two different strengthening-rates were introduced will be described in section 4.4.2. The above learning algorithm can also be seen as a simplified form of the following equation:

$$\tau_1 \frac{dW_{SA}(y_1, y_2)}{dt} + W_{SA}(y_1, y_2) = WS_2 \cdot \psi[A(y_2) \cdot T(y_1)], \qquad (4.1)$$

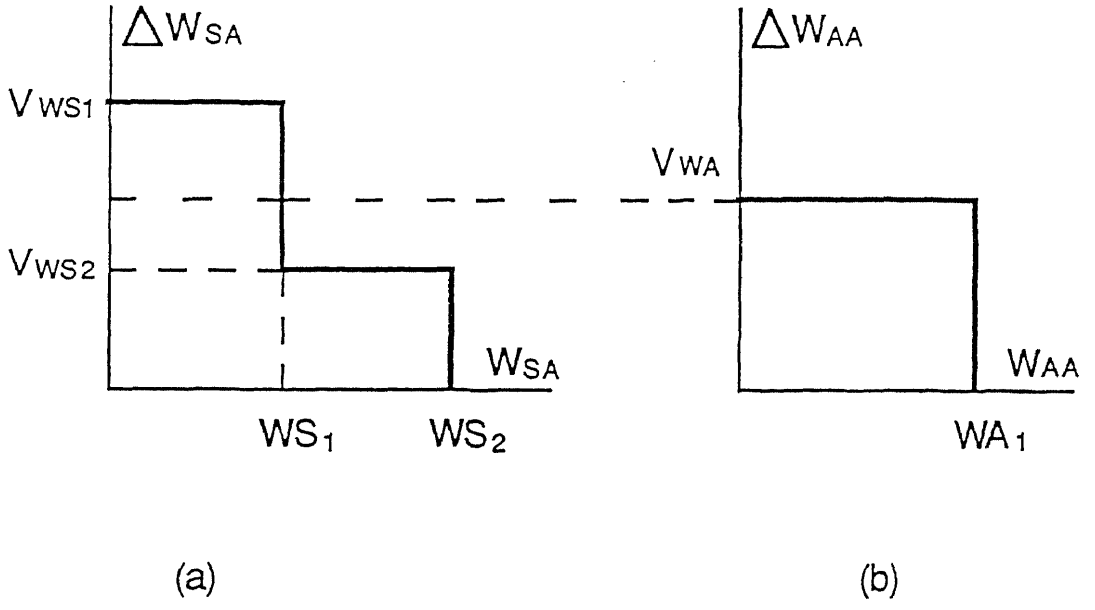


Fig. 4.3. Relationships between the strengthening-rates of the modifiable connections and their current strength in the learning process.

where

$$\psi[a] = \begin{cases} 1, & \text{if } a > 0 \\ 0, & \text{if } a \leq 0. \end{cases} \tag{4.2}$$

The element $S(y_1)$ is assumed to be an analog threshold element with time lag of the first order. The response of the element $S(y_1)$ is defined by

$$\mu_1 \frac{dS^*(y_1,t)}{dt} + S^*(y_1,t) = T(y_1,t) + \sum_y W_{SA}(y_1,y) \cdot A(y,t), \tag{4.3}$$

and

$$S(y_1,t) = \varphi_S[S^*(y_1,t)], \tag{4.4}$$

where

$$\varphi_S[a] = \begin{cases} S, & \text{if } a > S \\ a, & \text{if } 0 < a \leq S \\ 0, & \text{if } a \leq 0. \end{cases} \tag{4.5}$$

4.4.2 Inhibitory recurrent network

At the output of the heteroassociative network, both the current number word and the next words associated with it appear as a superimposed pattern. The inhibitory recurrent network is used to make the current number word suppresses the others so that the current word appears at the output of HASP.

Structure of the network is similar to that of the autoassociative memory models, but the modifiable connections denoted by $W_{AA}(y_1,y_2)$ are inhibitory, and the strength is gradually increased from an initial value, which is assumed to be zero, according to the following learning algorithm (L-4.2):

$$W_{AA}(y_1,y_2,t+\Delta t) = W_{AA}(y_1,y_2,t) + \Delta W_{AA}(y_1,y_2,t),$$

where $\Delta W_{AA}(y_1,y_2,t)$ is the strengthening-rate of connections $W_{AA}(y_1,y_2)$ at time t when a stimulus is presented through $T(y_1)$ and is determined as follows:

1. if $T(y_1,t) > 0$, $T(y_1,t) \cdot T(y_2,t) = 0$ and $W_{AA}(y_1,y_2,t) < WA$, then $\Delta W_{AA}(y_1,y_2,t) = V_{WA}$. Where, WA is a saturation value of $W_{AA}(y_1,y_2)$ and V_{WA} is some positive value.

2. otherwise, $\Delta W_{AA}(y_1,y_2,t) = 0$.

The relationship between the strengthening-rate of the modifiable connections and their current strength is shown in Fig. 4.3(b). With this algorithm the autocorrelation of a spatial pattern representing each number word is stored as a spatial distribution pattern of connections that have never been strengthened. If each number word is a binary pattern composed of 0 and 1 elements and at least one 1-element of the pattern is not included in the others, then the inhibitions between such 1-elements will form. Therefore, the multiple number words, which appear at the output of heteroassociative network simultaneously, compete through these inhibitions, and finally only the current number word appears at the output of HASP (cf. Hirai, 1983).

The time lag between the formation of associative relationships by $W_{SA}(y_1,y_2)$ and the formation of competitive function by $W_{AA}(y_1,y_2)$ can be considered as the origin causing the three phenomena describe in Section 4.2. In order to make the model yield this lag in learning, we introduced two different strengthing-rates in the learning algorithm (L-4.1) and made the saturation values and the strengthening-rates of the modifiable connections described in the two learning algorithms satisfy the following relations (L-4.3):

$$WS_1 < WS_2 < WA,$$

and

$$V_{WS_2} < V_{WA} < V_{WS_1}.$$

Therefore, the strength of $W_{SA}(y_1, y_2)$ can be guaranteed to be stronger than that of $W_{AA}(y_1, y_2)$ in the early stages of learning, and the time lag will arise. The changes in strength over time of the two modifiable connections are shown in Fig. 4.4.

The element $A(y_1)$ is assumed to be an analog threshold element with time lag of the first order. Response of the element $A(y_1)$ at time t is defined by

$$A(y_1, t) = \varphi[A^*(y_1, t)], \tag{4.6}$$

where

$$\varphi[a] = \begin{cases} a, & \text{if } a > 0 \\ 0, & \text{if } a \leq 0, \end{cases} \tag{4.7}$$

$$\mu_2 \frac{dA^*(y_1, t)}{dt} + A^*(y_1, t) = T(y_1, t) + W(y_1) \cdot S(y_1, t) - \sum_z W_{AA}(y_1, z) \cdot I(z, t), \tag{4.8}$$

and

$$I(z, t) = \varphi[A(z, t) - B(z, t)], \tag{4.9}$$

where $W(y_1)$ is a connecting coefficient from $S(y_1)$ to $A(y_1)$, and $B(z, t)$ is a response of the readout control unit described below.

4.4.3 Readout control unit

Transition between number words is performed by readout control units denoted by RCU in Fig. 4.2. As mentioned above, the next possible number words associated with the current one also appear at the output of the heteroassociative network, but they are suppressed by the current number word in the inhibitory recurrent

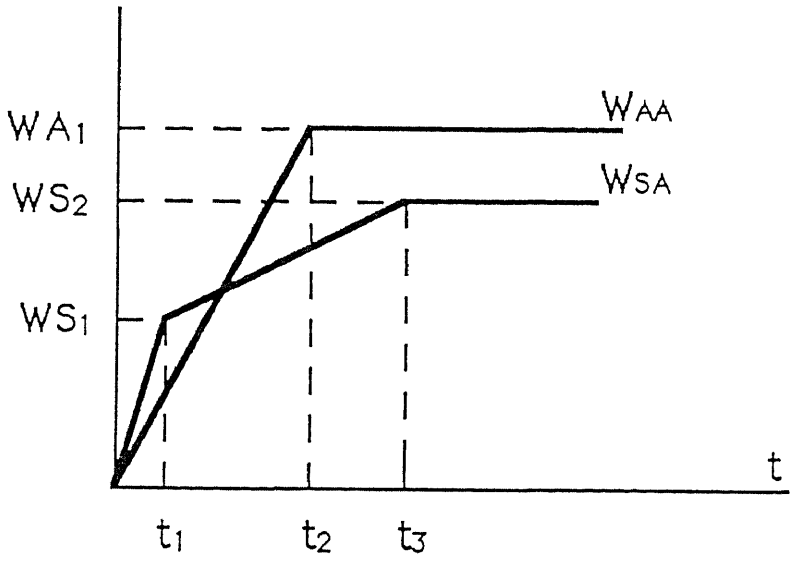


Fig. 4.4. Time course of the strength of the modifiable connections as learning proceeds

network. One of the next number words can be retrieved by suppressing the activity of the current one, because the suppresion releases the inhibition of the next number words by the current one. These next words begin to compete with each other in the inhibitory recurrent network and finally one of them wins the competition.

The structure of a readout control unit is shown in Fig. 4.5. A unit consists of two elements denoted by G and B in the figure. The G element performs a gating function as follows:

$$G(y,t) = rcs(t) \cdot \psi[rcs^*(t) \cdot A(y,t) + G(y,t-dt)], \tag{4.10}$$

where $rcs^*(t)$ is a positive differential signal of $rcs(t)$, which is a readout control signal referred by rcs below. The rcs^* becomes active for a brief period when rcs is activated. The argument $t - dt$ represents a time lag of the feedback loop from $G(y)$ to $G(y)$ which maintains the gated signal until rcs is inactivated. The output of the G element is fed into the B element, from which the output of the RCU comes. The response of the B element is defined by

$$\mu_3 \frac{dB(y,t)}{dt} + B(y,t) = G(y,t). \tag{4.11}$$

If $A(y)$ is active when rcs is activated, $G(y)$ becomes active by (4.10) and $B(y)$ begins to supress the response of $I(y)$. This supression causes the weakening of the competive power of a currently active number word and results in the activation of the next one.

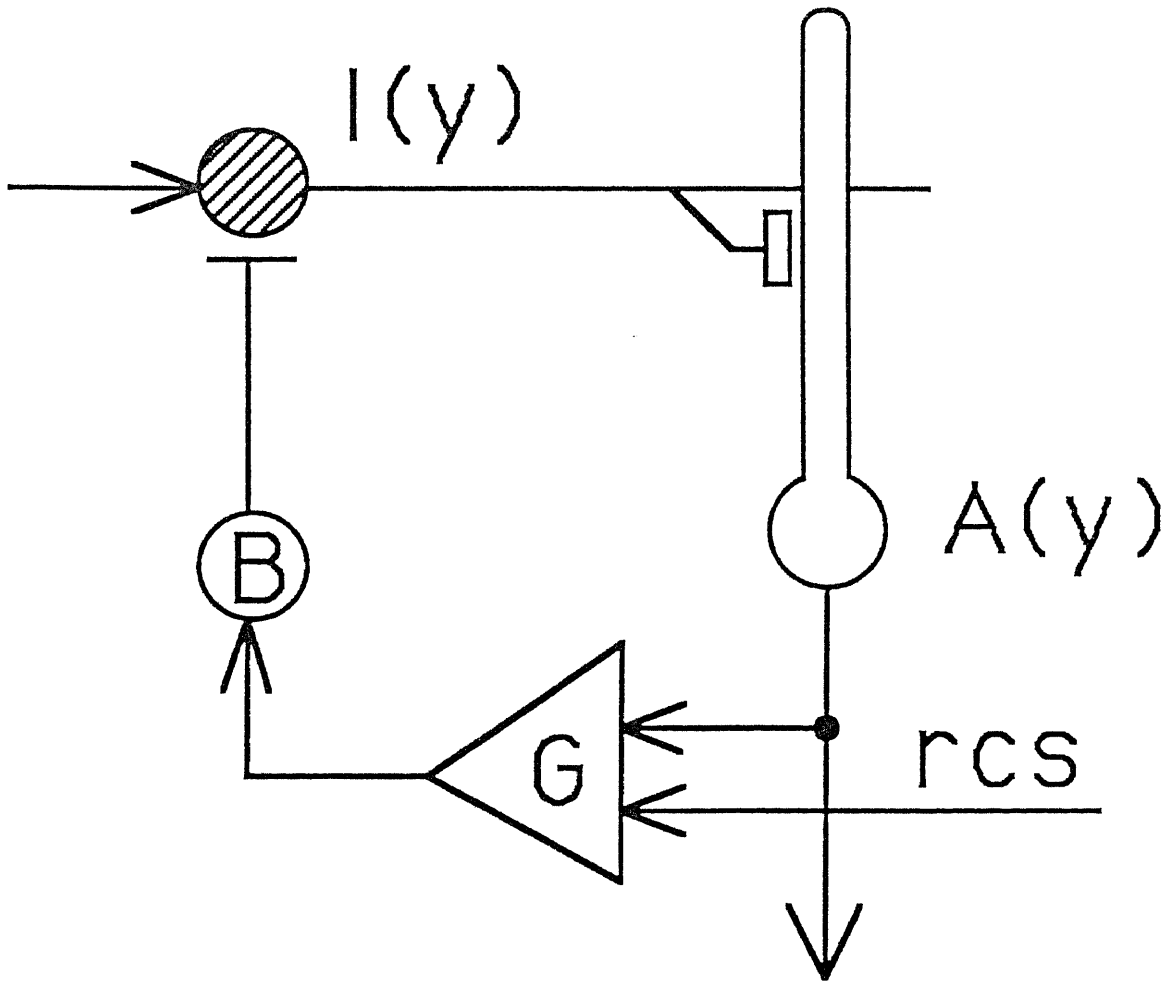


Fig. 4.5. Structure of a readout control unit. Network notations are the same as those in Fig. 2.1.

4.5 Computer Simulation

The learning and production of number word sequences by the model were simulated on a digital computer to examine whether the three phenomena described in Section 4.2 can be observed. In this section, we first describe how the network learns sequences, and then present the simulation results.

4.5.1 Learning of a sequence

When the network learns sequence "1, 2, 3, 4", for example, the patterns "1", "2", "3", "4" are presented in order through $T(y_1)$ shown in Fig. 4.2. The order of presentation is shown in Fig. 4.6. As shown in the figure each pattern is presented N times. After each pattern from "1" to "4" has been presented N times respectively, we say that the whole sequence has been learned one time.

While pattern "1" is being presented through $T(y_1)$, as described in Section 4.4.1, the excitatory connections $W_{SA}(y_1, y_2)$ are strengthened according to the learning algorithm (L-4.1) to store the associative relationship M["1"] = "1". At the same time, the inhibitory connections $W_{AA}(y_1, y_2)$ are also strengthened according to the learning algorithm (L-4.2) described in Section 4.4.2. After "1" has been presented N times, pattern "2" is presented. Since pattern "1" is appearing at the output of the network, it will be fed to the network through the positive feedback loops, and the associative relationship M["1"] = "2" will be established. Besides, pattern "2" will also appear at the output of the network soon, the associative relationship M["2"] = "2" will be established. At this time, if the whole sequence has been learned only a small number of times, since the strength of the

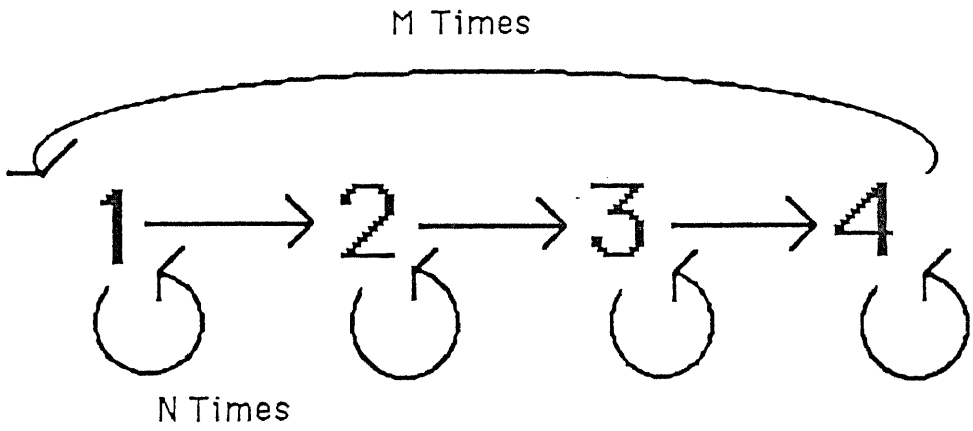


Fig. 4.6. Order of the presentation of the number word sequence.

inhibitory connections $W_{AA}(y_1, y_2)$ is weaker than that of the excitatory connections $W_{SA}(y_1, y_2)$ as shown in Fig. 4.4, the appearance of pattern "2" at the output of the network cannot suppress pattern "1", which is maintained by M["1"] = "1", through $W_{AA}(y_1, y_2)$. Therefore, a superimposed pattern of "1" and "2" appears at the output of the network while pattern "2" is being presented. On the other hand, if the whole sequence has been learned a large number of times, the strength of the inhibitory connections $W_{AA}(y_1, y_2)$ has become stronger than that of the excitatory connections $W_{SA}(y_1, y_2)$ as shown in Fig. 4.4. When the pattern "2" is presented, the pattern "1" disappears gradually because the pattern "2" supresses the pattern "1" through the inhibitory recurrent network such that finally only the pattern "2" appears at the output. Therefore, when the pattern "3" is presented at an early stage of learning, a superimposed pattern of "1", "2", "3" appears at the output of the network, and not only the associative relationships M["3"] = "3" and M["2"] = "3", but also the associative relationship M["1"] = "3" is established. On the other hand, when learning has proceeds, the pattern "1" disappears when the pattern "3" is presented, and the associative relationship M["1"] = "3" is not strengthened further (it will decay with time and finally disappear). Hence, the association M["2"] = "3" gradually becomes stronger than M["1"] = "3" as the learning proceeds. Moreover, by the presentation of the pattern "4", M["4"] = "4", M["2"] = "4", and M["3"] = "4" can be established. Also, the association M["1"] = "4" may be established with very weak strength.

We can sumerize the learned results described above as follows. Through the learning of the sequence "1, 2, 3, 4" in the way shown in Fig. 4.6, the associa-

tive relationships from the each pattern to itself: M["1"] = "1", M["2"] = "2", M["3"] = "3", M["4"] = "4" and the associative relationships: M["1"] = "2", M["1"] = "3", M["2"] = "3", M["2"] = "4" are established in the modefiable connections $W_{SA}(y_1, y_2)$. Besides, the associative relationship M["1"] = "4" may be established with very weak strength. If the whole sequence has been learned only a small number of times, the strength of M["1"] = "2" (or M["2"] = "3") is stronger than that of M["1"] = "3" (or M["2"] = "4") only a little. But as the learning proceeds, the strength of the former association becomes stronger than that of the latter one.

4.5.2 Parameters of the model

The parameters of the network used in the computer simulation are as follows. The time constants of the S, A and B elements, μ_1, μ_2, μ_3, are set to 1.0, 2.0 and 2.0 time units of numerical calculation. The response saturation value of the elements $S(y_1)$, S, is set to 5.0. The strengthening-rates of the modifiable excitatory connections $W_{SA}(y_1, y_2)$, V_{WS_1} and V_{WS_2}, described in learning algorithm (L-4.1) are set to 0.028 and 0.005. The positive value WS_1 and the saturation value WS_2 described in learning algorithm (L-4.1) are set to 1.2 and 2.5 respectively. The strength of all the connections $W_{SA}(y_1, y_2)$ are reduced by 0.0001 in each presentation of a pattern until it reaches its saturation value WS_2. The strengthening-rate of the modifiable inhibitory connections $W_{AA}(y_1, y_2)$, V_{WA}, described in learning algorithm (L-4.2) is set to 0.012. The saturation value, WA, is set to 3.5. The connecting coefficient $W(y_1)$ from $S(y_1)$ to $A(y_1)$ is set randomly to a value between 1.0 and 1.2. When a sequence is learned, each element of the sequence is presented to the network

through $T(y_1)$ 20 times.

4.5.3 Results of simulation

4.5.3.1 Simulation on the three portions

The learning and the production of the sequence consisting of 4 number words, "1", "2", "3", and "4", were simulated to examine whether the sequences produced by the network change from the unstable structure to the conventional structure via the stable nonconventional structure as the learning proceeds. The patterns representing "1", "2", "3" and "4" were assumed to be arrays composed of 4 elements as follows: "1"=(1, 0, 0, 0), "2"=(0, 1, 0, 0), "3"=(0, 0, 1, 0) and "4"=(0, 0, 0, 1). Since the connecting coefficient $W(y_1)$ from $S(y_1)$ to $A(y_1)$ shown in Fig. 4.2 is set randomly to a value between 1.0 and 1.2, ten groups of random numbers were produced by the computer. The learning and production of the sequence by the network were simulated for each group, which is set to $W(y_1)$ and is assumed to correspond to a given child. The simulation results for all the ten groups of random numbers are shown in Table 4.3. From this table we can see that if the sequence has been learned only a small number of times (less than about 4 times), the sequence produced is unstable; if the number of presentations increses but remains less than about 9, the sequence produced becomes stable but nonconventional; and if it is learned furthermore, the sequence produced becomes stable and conventional. In addition, it should be noted that in the stable nonconventional portions, both the case that "2" is skipped and the case that "3" is skipped exist. Which number

Table 4.3. Simulation result on the three portions of sequence

Groups of random number \ Learning times	Unstable portions			Stable nonconventional portions					Conventional portions			
	1	2	3	4	5	6	7	8	9	10	11	12	13
1		3–4	3–4	1–3–4	1–3–4				1–3–4	1–2–3–4			
2		4	3–4	3–4	1–3–4					1–3–4	1–2–3–4		
3		4	3–4	1–4	1–4	1–3–4	1–2–3–4						
4		4	3–4	3–4	1–3–4	1–2–4	1–2–4	1–2–3–4					
5		3–4	1–3–4	1–3–4							1–3–4	1–2–3–4	
6		4	2–4	2–4	1–2–4	1–2–4	1–2–3–4						
7		4	4	1–4			1–4	1–2–4	1–2–4	1–2–4	1–2–3–4		
8		3–4	3–4	3–4	1–2–3	1–2–3	1–2–3	1–2–3–4					
9		3–4	4	1–3–4	1–2–4				1–2–4	1–2–3–4			
10		4	4	2–4	1–2–4			1–2–4	1–2–3–4				

word is skipped depends on the group of random numbers that set the connections $W(y_1)$.

Why our network can produce the sequences of three portions is analyzed as follows. As an example, Table 4.4 shows the strength of the excitatory connections $W_{SA}(y_1, y_2)$ after the sequence is learned 3, 4, 7, 8, 15, and 30 times when the 1st group of the randfom number {1.161,1.001,1.025,1.065} initializes the connections $W(y_1)$. From this table, we can see that the change of the strength of the connections $W_{SA}(y_1, y_2)$ at the early stages of learning (see (a) and (b) of Table 4.4) is much larger than that after the sequence has been learned many times (see (c) and (d) of Table 4.4). Therefore, at the early stages of learning, the sequence produced may easily vary from trial to trial, but become stable as the learning proceeds as shown in Table 4.3. If we use a function $S[a]$ to express the strength of an associative relationship "a", from (c) of Table 4.4 we can see that $S_c[M(\text{“1”}) = \text{“2”}] = W_{SA}(4,2) = 1.623$ and $S_c[M(\text{“1”}) = \text{“3”}] = W_{SA}(4,3) = 1.508$, where the subscript c specifies that the strength of the above two associative relationships is in the case of (c) shown in Table 4.4. From (e) of Table 4.4, we can see that $S_e[M(\text{“1”}) = \text{“2”}] = 2.274$ and $S_e[M(\text{“1”}) = \text{“3”}] = 1.709$, and from (f) $S_f[M(\text{“1”}) = \text{“2”}] = 2.5$ and $S_f[M(\text{“1”}) = \text{“3”}] = 1.589$. From the above we can see that $S_c[M(\text{“1”}) = \text{“3”}]$ is closest to $S_c[M(\text{“1”}) = \text{“2”}]$ comparing to the case of (e) and (f). In case (c), the sum of the strength of the connections $\sum_y W_{SA}(y, 3)$, which is used to obtain the response of pattern "3", is stronger than that of $\sum_y W_{SA}(y, 2)$, which is used to obtain the response of pattern "2". In addition, the strength of connection $W(3)$, through which the 1-element of pattern "3"

Table 4.4. Matrices of the strength of $W_{SA}(y_1, y_2)$

0.0	0.0	0.0	1.274
0.245	0.554	1.282	1.224
0.825	1.285	1.247	0.586
1.273	1.255	0.996	0.306

After 3 times of learning

(a)

0.028	0.554	0.556	1.268
0.629	1.106	1.374	1.316
1.214	1.377	1.339	1.138
1.360	1.347	1.247	0.298

After 4 times of learning

(b)

0.028	1.034	1.092	1.622
1.218	1.444	1.650	1.592
1.380	1.653	1.615	1.463
1.621	1.623	1.508	0.806

After 7 times of learning

(c)

0.028	1.026	1.084	1.699
1.210	1.436	1.737	1.684
1.372	1.740	1.707	1.505
1.708	1.715	1.600	1.022

After 8 times of learning

(d)

0.028	0.970	1.028	2.268
1.154	1.380	2.281	2.038
1.316	2.314	2.191	1.499
2.317	2.274	1.709	0.966

After 15 times of learning

(e)

0.028	0.850	0.908	2.500
1.034	1.260	2.500	2.500
1.195	2.500	2.500	1.379
2.500	2.500	1.589	0.846

After 30 times of learning

(f)

passes, is larger than that of $W(2)$, through which the 1-element of pattern "2" passes. Therefore, "2" is skipped and "3" is produced next to "1" by M["1"] = "3". But as the learning proceeds, because $S[M(\text{"1"}) = \text{"2"}]$ becomes much larger than $S[M(\text{"1"}) = \text{"3"}]$ (see the case of (e) and (f)), "2" is produced always next to "1", and the sequence produced becomes conventional.

The learning and production process of a sequence consisting of 12 number words from 1 to 12 was also simulated. A simulation example of repeated production of the sequence for one group of random numbers is shown in Table 4.5. In this simulation example, the sequence was divided into three groups to be learned. The first sequence portion consisting of "1, 2, 3, 4" was learned 30 times, the second portion consisting of "5, 6, 7, 8" was learned 9 times, and the third portion consisting of "9, 10, 11, 12" was learned from 2 to 5 times. The sequences shown in Table 4.5 are the cases in which the third sequence portion was learned 2, 3, 4, and 5 times respectively. It is evident that the structure of the sequence produced by the network is similar to that produced by a child, as shown in Table 4.1. In Table 4.5, the stable conventional sequence protion is "1, 2, 3, 4", the stable nonconventional portion is "5, 7, 8", and the unstable portion consists of the words following "9".

4.5.3.2 Simulation on the omission of "fifteen"

We made the network learn the number word sequence "4, 5, 6, 7" 30 times at first, and then made it learn the sequence "14, 15, 16, 17" to see whether "15" would be omitted more often than all other words when the production of the sequence is

Table 4.5. Simulation on production of the sequence including the three portions for one group of random number

1 → 2 → 3 → 4	→ 5 → 7 → 8 → 12	
1 → 2 → 3 → 4	→ 5 → 7 → 8 → 11 → 12	
1 → 2 → 3 → 4	→ 5 → 7 → 8 →	9 → 11 → 12
1 → 2 → 3 → 4	→ 5 → 7 → 8 →	9 → 10 → 12
Conventional portions	Stable nonconventional portions	Unstable portions

stable but nonconventional. In order to exclude the bias of the connections $W(y_1)$ from the production of the number words, all $W(y_1)$ were set to the same value, 1.0. The patterns representing the number words were assumed to be two-dimensional binary (0 and 1) arrays composed of 3×3 elements as shown in Fig. 4.7. From this figure we can see that except the pair "5 and 15", which has no common part, all the other pairs "4 and 14", "6 and 16", and "7 and 17" have common parts. That is, only the word "15" is of an irregular pattern. There was some amendment to the other parameters of the network, since the encoding of the number words used here is different from that in Section 4.5.3.1.

The simulation result shows that when the number of learning trials of the whole sequence "14, 15, 16, 17" is smaller than 23, the production of the sequence is unstable; when the number of learning trials is between 23 and 28, the production is stable but "15" is omitted always; and when the learning trials increases, the sequence produced becomes conventional.

4.5.3.3 Simulation on the recitation context study

All parameters of the network used in this simulation were the same as those presented in Section 4.5.3.1. Sequence "1, 2, 3, 4" was used as an example and the representation of the number words was also the same as those in Section 4.5.3.1. The porpose of the simulation was to see how many steps in the numerical simulation are necessary to retrieve pattern "3" by supplying "2" and by supplying a superimposed pattern of "1" and "2" to the network. The simulation was carried

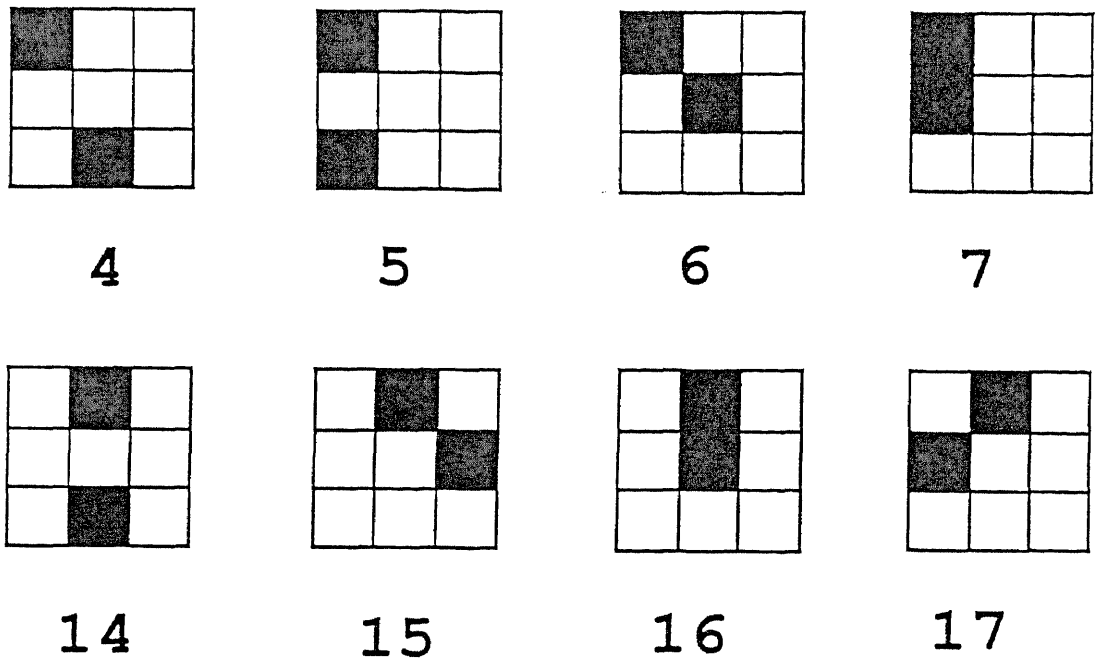


Fig. 4.7. Patterns representing the number words

out for all ten groups of random numbers initializing the connections $W(y_1)$, and the result is shown in Table 4.6. The retrieval steps were measured at three learning stages: the stable nonconventional portions, the first part of conventional portions, and the final part of conventional portions. The notation "no" in the table indicates that the pattern "3" could not be retrieved, and the numbers express the number of steps which are necessary to retrieve the pattern "3". From this table we can see that, in the stable nonconventional portion, the next word "3" cannot be retrieved by providing only one pattern "2" but can be retrieved by providing the superimposed pattern of "1" + "2"; In the first part of conventional portions, in almost all cases the number of steps necessary to retrieve "3" is greater for the pattern "2" than for the superimposed pattern; and in the final part of conventional portions, in almost all cases the number of steps necessary to retrieve "3" is the same for both the pattern "2" and the superimposed pattern "1"+"2". That is, in the early stages of learning, the production of the next pattern is facilitated by prompting with two, rather than one, preceding patterns.

4.6 Summary

We have described a neural network model which can mimic some aspects of the learning process of sequence, and have used the model to present an explanation for the three psychological phenomena occurring in the learning of counting by children. In order to construct the model, the learning algorithms of an earlier model of human associative processing, HASP, was modified to enable the learning and production of sequences by the network to be simulated on a digital computer.

Table 4.6. Simulation result on a recitation contex study

Groups of random number	Stable nonconventional portions			First part of conventional portions			Final part of conventional portions		
	Input "2"	Input "1"+"2"	Difference	Input "2"	Input "1"+"2"	Difference	Input "2"	Input "1"+"2"	Difference
1	No	10	–	8	7	1	7	7	0
2	No	14	–	7	7	0	7	7	0
3	No	no	–	15	12	3	7	7	0
4	No	10	–	10	8	2	7	7	0
5	No	26	–	7	6	1	7	6	1
6	No	10	–	11	11	0	8	8	0
7	No	10	–	9	9	0	7	7	0
8	No	no	–	9	no	–	8	8	0
9	No	12	–	8	10	– 2	8	7	1
10	No	10	–	20	9	11	8	8	0

The simulation results show that, (1)the sequences produced by the network are also of the same typical structure as that produced by children; (2)it is much more difficult for the network, as it is for children, to learn irregular number words; and (3)as with children, in the early stages of learning the production of sequences by the network depends on the recitation context.

Chapter 5

Conclusion

In this dissertation, we have successfully represented three kinds of cognitive processes, which are the problem solving, knowledge representation, and learning, with neural networks and demonstrated that all these networks can be constucted within the framwork of the associative network, HASP.

The research was based on the belief that the memory is the most fundamental and essential function in human cognitive processes: To solve a problem, the related information and knowledge must be effectively and contextually retrieved from memories; To learn to perform a new cognitive task, the old related knowledge must be well utilized and the new acquired knowledge must be memorized; A key problem of knowledge processing is how to effectively and quickly access the necessary concepts or information from large bodies of knowledge base.

This dissertation has described how the various kinds of memories are constructed and how they are properly combined to performing the cognitive tasks. We have presented a model that has a general structure for problem solving, a new scheme for knowledge representation, and a new method to explain psychological

phenomena. We can say that the research is very novel and meaningful as an approach to the understanding of the architecture of the brain and the mystery of human cognitive abilities.

As with any new approach, directions of future research are nearly limitless. In the model for problem solving, the visual system and the output system were assumed to exist, and these present immediate work to pursue. Moreover, the cognitive tasks chosen in the dissertation are very simple and the three cognitive processes, problem-solving, knowledge representation, and learning, are considered independently. In our brains, however, a large number of cognitive tasks are more complex and the various kinds of cognitive processes must be working together. For example, when people do something, the necessary information or concepts must usually be very quickly and effectively retrieved from large bodies of knowledge base and learning may usually occur to improve the performance or to acquire new knowledge and skills. To apply the proposed models to much more complex tasks by combining them properly are left as a future work. From our works, however, we can say that the neural networks can shed a much more light on the modeling of the brain functions than the symbolic approach and is very promising.

ACKNOWLEDGEMENTS

I owe a large debt of gratitude to many people for the completion of this dissertation.

I first would like to thank my research supervisor, Dr. Yuzo Hirai. The works included in the dissertation were carried out under his supervision and were greatly influenced by his enthusiasm and insights.

I would also like to acknowledge Professor Kazuaki Ando, Professor Shuichi Itabashi, Professor Ryo Natori, and Dr. Masumi ishikawa for their helpful comments on my works.

In addition, I am grateful to Professor Kenji Hiwatashi for his encouragement during the research.

Finally, I would like to thank every person who has been or is working with me in the visual information processing laboratory, because they provided me with a nice enviroment in which a successful research was possible.

REFERENCES

Anderson, J. R. *Cognitive psychology and its implications.* Freeman, San Francisco, 1980.

Anderson, J. R. *The architecture of cognition.* Harvard University Press, 1983.

Anderson, J. R. & Bower, G.H. *Human associative memory.* Washington, D.C.: V.H. Winston, 1973.

Anderson, J. A. & Hinton, G. E. Models of information process in the brain. In G. E. Hinton and J. A. Anderson (Eds.) *Parallel models of associative memory*, Lawrence erlbaum associates, Publishers. Hillsdale, New Jersey, 1981.

Brachman, R. J. What 'IS-A' is and isn't. *Proceedings of Canadian Society for Computational Studies of Intelligence-82*, Canada, 212-220, 1982.

Brachman, R. J. I lied about the trees. *AI magazine* 6(3), 80-93, 1985.

Brown, J. S. & Burton, R. R. Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science* 2, 155-192, 1978.

Brown, J. S. & VanLehn, K. Repair theory: a generative theory of bugs in procedural skills. *Cognitive Science* 4, 379-426, 1980.

Cox, L. S. Diagonosing and remediating systematic errors in addition and subtraction computations. *Arithmatic Teacher* 2, 151-157, 1975.

Etherington, D. W. & Reiter, R. On inheritance hierarchies with exceptions. In M. Ginsberg (Ed.), *Nonmonotonic resoning.* Los ALTOS, California, 1987.

Fahlman, S. E. *NETL: A system for representing and using real-world knowledge.* Cambridge, MA: MIT Press, 1979.

Fahlman, S.E., Touretzky, D.S., and Van Roggen, W. Cancellation in a parallel semantic network. *Proc. IJCAI-81*, Vancouver, B.C., 257-263, 1981.

Fahlman, S. E. Three flavors of parallelism. *Proceedings of Canadian Society for Computational Study of Intelligenc-82*, Canada, 230-235, 1982.

Fukushima, K. *Neural networks and self-organization.* Kyoritsu Publishing Co., Ltd., 1979. (in Japanese)

Fukushima, K. *Neural Networks and Information Processing.* Asakura Publishing Co., Ltd., 1989. (in Japanese)

Fuson, K. C., Richards, J., and Briars, D. J. The acquisition and elaboration of the number word sequence. In C. Brainerd (Ed.) *Progress in logical development: children's logical and mathematical cognition*, vol. 1. Springer, Berlin Heidelberg, New York, 33-92, 1982.

Greeno, J. G. A study of problem solving. In R. Glaser (Ed.) *Advances in instructional psychology*, vol. 1, Erlbaum, Hillsdale, 13-75, 1978.

Ginsburg, H. P. *The development of mathematical thinking.* Academic Press, New York, London, Tokyo, 1983.

Hinton, G. E. Implementing semantic networks in parallel hardware. In G. E. Hinton & J. A. Anderson (Eds.), *Parallel Models of Associative Memory.* Lawrence erlbaum associates, Publishers. Hillsdale, New Jersey, 1981.

Hirai, Y. A model of human associative processor (HASP). *IEEE Trans.* SMC-13, 5, 851-857, 1983.

Hirai, Y. An orthogonal learning network incorporating HASP associative network structure. *Trans. of IECE of Japan*, J67-A, 1154-1161, 1984. (in Japanese)

Hirai, Y. Representation of procedural knowledge by a model of human associative processor, HASP. *Trans. of IECE of Japan*, J69-D, 1743-1753, 1986. (in Japanese)

Hirai, Y. & Ma, Q. Representation of problem solving processes by a model of associative processor, HASP – A case study of the process of addition. *Trans. of IECE of Japan*, J70-D, 983-994, 1987. (in Japanese)

Hirai, Y, & Ma, Q. Associative networks underlying problem-solving: Representation of the process of addition by a model of an associative processor, HASP. *Proceedings of the IEEE First Annual International Conference on Neural Networks*, II.33-40, 1987.

Hirai, Y. & Ma, Q. Modeling the process of problem-solving by associative networks capable of improving the performance. *Biol. Cybern.* 59, 353-365, 1988.

Hopfield, J. J & Tank, D. W. "Neural" computation of decisions in optimization problems. *Biol. Cybern.* 52, 141-152, 1985.

Kintsch, W & Greeno, J. G. Understanding and solving word arithmetic problems. *Psychological Review*, vol. 92, No. 1, 109-129, 1985.

Kohonen, T. Correlation matrix memories. *IEEE Transactions on computers*, vol. c-21, No. 4, 353-359, 1972.

Kohonen, T. *Associative Memory.* In K. S. Fu et al. (Eds.) Springer-Verlag, Berlin, Heidelberg, New York, 1978.

Ma, Q. Representation of problem-solving processes by a model of associative processor, HASP – A case study of the process of addition. *Master's thesis*, 1987.

Ma, Q. & Hirai, Y. Modeling the acquisition of counting with an associative network. *Biol. Cybern.* 61, 271-278, 1989.

Marr, D. *Vision*, Freeman, 1982.

Newell, A & Simon, H. A. *Human problem solving.* Prentice Hall, 1972.

Quillian, M. R. The teachable language comprehender. *Communications of ACM*, 12, 8, 459-475, 1969.

Reiter, R. A logic for default reasoning. *Artificial Intelligience* 13, 81-132, 1980.

Rosenblatt, F. *Principles of neurodynamics.* Washington, D. C.: Spartan Books, 1962.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature*, vol. 323, 533-536, 1986.

Rumelhart, D. E. & McClelland, J. L. On learning the past tenses of English verbs. In J. L. McClelland, D. E. Rumelhart and the PDP Research Group (Eds.) *Parallel and distributed processing*, vol.2. MIT Press, Cambridge, 216-271, 1986.

Squire L. R. & Cohen N.J. Human memory and amnesia. In G. Lynch, J. L. McGaugh, N. M. Weinberger (Eds.) *Neurobiology of learning and memory.* Guilford Press, New York, 1984.

Tulving, E. *Elements of episodic memory.* Oxford: Clarendon Press, 1983.

Wang, N. C. & Norman, D. A. Primary memory. *Psychological Review*, vol. 72, No. 2, 89-104, 1975.

Young, R. M. & O'Shea, T. Errors in children's subtraction. *Cognitive Science* 5, 153-177, 1981.