A Hybrid Rule-based System

-- How Variables are Involved in Connectionist Rule-based Systems --

Fukumi Kozato
Department of Electronic Engineering, Tokyo Institute of Polytechnics
1583 Iiyama, Atsugi-shi, Kanagawa, 243-02, JAPAN
tel & fax: 0462-42-9567, fukumi@ee.t-kougei.ac.jp

Abstract

Modeling of a connectionist rule-based systems (or Neuro-AI hybrid system) discussed through the paper will be a fruitful step towards the practical modeling of human cognition. This paper investigates a plausible and useful integration method of symbolic AI techniques and connectionist models and proposes a practical implementation, mainly how variables can be included in the structured information provided as facts and rules in the system.

1 Introduction

Although connectionist models and symbolic AI techniques are often seen as rivals, the aim of this paper is to show the usefulness of a hybrid system of symbolic AI and connectionist models, namely a rule-based connectionist inference system. The system inherits aspects of two previously proposed hybrid systems [Kozato&DeWilde,91ab], and combines their advantages. The first system handles probabilistic knowledge represented only as propositions, whereas the second handles structured information or predicates in a very limited way. The system described here handles structured information with similarity between the individual components, involves lenient variable handling to allow flexible inferences, and possesses an efficient rule selection or search mechanism by evaluating several units of knowledge together. In addition to these functions, strict variable handling mechanism is also included. To yield these functions together, the system is designed as a combination of four different types of connectionist networks and a data buffer. The networks work in cooperation to derive new facts and the data buffer keeps the facts provided by the user or derived by the networks.

1.1 Representing Information Units and Rules

Each unit of knowledge provided by the user and the facts derived by the system are both represented as information units. An information unit consists of three information fragments: a subject concept, an object concept, and the relation. Strict (Syntactic) variables or just variables can be included in information units in the same manner as a subject or object concept. For example, a variable X is described in two forms: s:X which can be substituted with a subject concept and o:X with an object concept. An information unit with a variable looks as follows:

like(s:X,o:bananas)

A rule implemented in the system consists of a pair of premises followed by a conclusion. The premises and conclusions are expressed as information units. For example, a rule looks like the following:

rule1: IF like(s:gorillas,o:X) and belong(s:X,o:fruits)
THEN eat(s:gorillas,o:X)

The premises of this rule can be satisfied with pairs of information units such as

like(s:gorillas,o:bananas) and belong(s:bananas,o:fruit), or
like(s:gorillas,o:apples) and belong(s:apples,o:fruit)

The former pair derives eat(s:gorillas,o:bananas) and the latter eat(s:gorillas,o:apples).

As mentioned above, the system is equipped with a lenient variable handling mechanism. For example, the premise part of rule1 may match like(s:chimps,o:bananas) when like(s:gorillas,o:bananas) is absent in the data buffer, providing that s:gorillas and s:chimps are represented as very similar information fragments in the system inference domain. In such a case s:gorillas of rule1 can be called a lenient variable or semantic variable and the inference operation which involves such variables is called semantic inference.

1.2 Distributively Represented Symbols and Loose Pattern Matching

There are basically two ways to represent symbols in a connectionist network. One is the localised method by which only one symbol is allocated in a single unit, and the other is the distributed method by which a symbol is assigned to a collection of units and each unit may represent fractions of more than one symbol. The model proposed in

this paper mainly take the latter method in a Hopfield binary neural network because it seems the way to yield the maximum capability of the connectionist model and brings several advantages as follows:

(1) Even with the binary state units (on or off), the system is capable of expressing graded certainty of information.

(2) The system may gain a high capacity to store knowledge with fewer units.

(3) Similarity information between units of knowledge is soundly implemented.

Loose pattern matching, which is one of the fundamental properties of many connectionist models, works very effectively if it participates in a rule-based inference operation. It has been utilised for a rule selection mechanism and the interpretation of vague knowledge.

1.3 Lenient Variable Handling

Loose pattern matching governs a certain kind of variable handling mechanism with no additional structure or facility. Unlike strict variable handling employed by ordinary symbol processing systems, lenient variable handling is based on the semantic similarity of information.

The model described in this paper represents information in a Hopfield binary neural network to realise lenient variable handling in the following manner:

(1) Units of information handled in the systems are represented as structured information composed of subject/object concepts and relations.
(2) Subject/object concepts are provided with a similarity degree to the other concepts, e.g. s:chimps =50%= s:gorillas.
(3) Each subject/object concept is allocated to a collection of neurons, and the neuron patterns for the concepts overlap in proportion to the similarity degree between the concepts, e.g. for the above similarity information a half of the fractions of the information fragments s:gorillas and s:chimps are allocated to the same neurons.

In this model, s:gorillas is called a lenient variable when a network training pattern representing like(s:gorillas,o:bananas) accepts a unit activation pattern like(s:chimps,o:bananas).

Lenient variables are quite suitable for the plain architecture of connectionist networks. This is because no additional structure or facility is required for the Hopfield network for the introduction of variables to the inference operation. Besides that, the generalisation function of connectionist models can be used to perform approximate inferences. Owing to the generalisation effect, any network training pattern can match a similar but not exactly identical unit activation pattern. A subject or object concept distributively represented in the Hopfield network may work as a lenient variable for a subject or object concept included in the initial input information, if they are very similar, or in other words if their representations in the Hopfield network overlap a lot.

Similar models previously proposed have not taken the idea of lenient variables into account. One but a very clear reason is that most systems of this kind are based on the localist representation method. The typical examples are the models proposed by Ballard [Ballard, 86], by Shastri [Shastri, 88], by Shastri and Ajjanagadde [Shastri & Ajjanagadde,89], and by Hendler [Hendler,91]. It seems very troublesome that a concept is leniently represented by a localist method. There is a model which takes a distributed method to represent information, that is the model proposed by Touretzky and Hinton [Touretzky&Hinton,88]. However, it has not expanded the generalisation effect towards the new interpretation of variables. More discussion can be found in [Kozato, 93].

2 Overview

The system consists of 5 parts as described in Figure1.

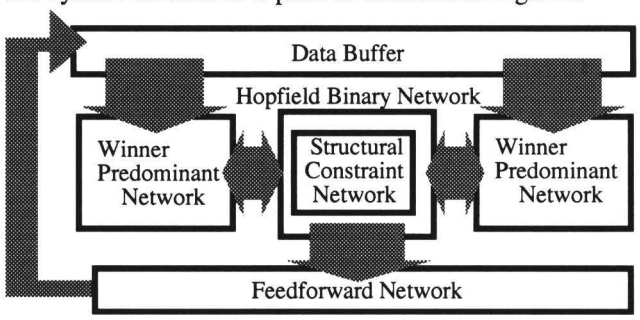

Figure 1 System Overview

The flow of information in the system is presented by the activation signals. The data buffer hands over certain information units to the winner predominant networks and receives additional information units derived by the feedforward network. During the update phase, the Hopfield binary network receives as well as sends back some information units from/to the winner predominant networks in the form of information fragments, and also sends out and receives information fragments to/from the structural constraint network. After this phase, the Hopfield network hands over the information fragments left in it to the feedforward network. The sequence of an inference operation proceeds as follows:

<Step1> The data buffer sends out activation signals to the winner predominant networks in order to project certain information units called the initial input information onto them.
<Step2> Each of the winner predominant networks passes the information units it holds to the Hopfield binary network by activating the units to which the information fragments included in these information units are allocated.
<Step3> The Hopfield network starts the unit update

process in order to select a rule by which a new information unit can be derived.

<Step4> After certain steps of the update process of Hopfield network units which is called an update operation term, the Hopfield network communicates with the winner predominant networks to exchange information so as to adjust the direction of the update operation toward a proper stable unit activation pattern which must express two information units as the premises of a rule.

<Step5> During the update operation phase which consists of a number of update operation terms, the Hopfield network also sends out activation signals to the structural constraint network to solve certain structural constraints between two information units as the premises of a rule when they include the same variables.

<Step6> An update operation phase terminates when only one information unit is left in each winner predominant network and it cannot be changed by a further update operation term. This termination indicates that the Hopfield network has selected a rule which can be used to derive a new information unit. Then the information fragments and variables left active in the Hopfield network are sent out to the input port of the feedforward network to derive a new information unit.

<Step7> Through the feedforward network, an information unit is generated and sent out from the output port. If the premises and conclusion of the selected rule include the same variable, then variable substitution may also take place.

<Step8> The information unit derived from the feedforward network is then handed over to the data buffer to be stored and used for another inference operation.

3 Architecture

3.1 The Data Buffer

The data buffer is used to store all the information units of a certain domain provided by the system user and derived through the inference operation. When the system starts the inference operation, the buffer hands over a certain number of information units to the winner predominant networks as the initial input information, and also when the system ends the inference operation, it accepts a new information unit which has just been derived.

3.2 The Hopfield Binary Network

The Hopfield binary network is used as a loose pattern matching machine to select one of the rules implemented in the system. The rule selection is completed by choosing a rule whose premises are most likely to be satisfied by the initial input information projected on the winner predominant networks. An actual pattern matching process performed by the Hopfield network is completed in a sufficient number of the update operation terms. One update operation term is made up of a certain number of asynchronous unit updates, e.g., the total number of units of the Hopfield network.

3.2.1 Processing Units

Hopfield network units have only two states, on or off, and are categorised into the following three units:

(1) Concept units: The indiscriminately divided fractions of an information fragment are allocated to a collection of concept units, and the information fragment is called the base of those units. Because each concept unit represents a concept fractionally, the entire set of concept units with the same base being active would represent a concept with 100% validity.

(2) Relation units: Each relation is allocated to a relation unit.

(3) Variable units: Each variable is allocated to a variable unit.

3.2.2 Structure

The network is divided into two bodies: one is called the left bank and the other the right bank. Each bank which comprises a set of concept units, relation units and variable units is used to implement one of the two premises of a rule. Within a bank, all the concept units are fully connected to all the relation units, and all the relation units are fully connected to all the variable units. However, there are no direct connections between concept units and variable units since they are semantically independent of each other. Between the two banks, all the relation units are fully connected. This enables two rule premises in each bank to be recalled as one pattern after the update operation phase.

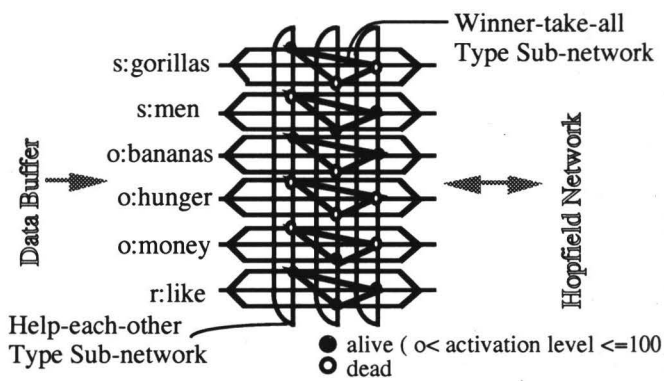

Figure 2 A Winner Predominant Networks and Live Sub-Networks

3.3 The Winner Predominant Networks

The winner predominant networks maintain the original combinations of information fragments as autonomous information units. Because each information unit consists of a set of information fragments, it is essential to ensure correct combination so as to represent correct information.

Each network comprises two kinds of sub-networks called the help-each-other type sub-networks and the winner-take-all type sub-networks as illustrated in Figure2. They consist of the nodes and bidirectional links between them. The necessary number of nodes is varied but always

a multiple of the total number of the information fragments included in the system inference domain. Each node has two states, alive or dead. A node becomes alive if it is activated at the projection of the initial input information from the data buffer, otherwise it is dead. Such a node is called a live node. Only live nodes can be operational and possess a certain level of activation signal between inactive and fully active. Only the sub-networks which include live nodes take part in the inference operation. They are called live sub-networks.

3.3.1. Help-each-other Type Sub-networks

A help-each-other type sub-network enhances the unity of an individual information unit. It chains nodes to which different information fragments are allocated. The connections are made with bidirectional links so that a node receives as well as sends out an activation signal from and to the adjacent nodes.

A sub-network represents the existence of an information unit as the three live nodes to which either a subject concept, an object concept or a relation is allocated. Such live nodes work cooperatively within a sub-network to keep themselves active by sending encouragement signals to each other. The magnitude of the encouragement signal is in proportion to the activation level of the live nodes. Figure3 describes the activity of a help-each-other type sub-network.

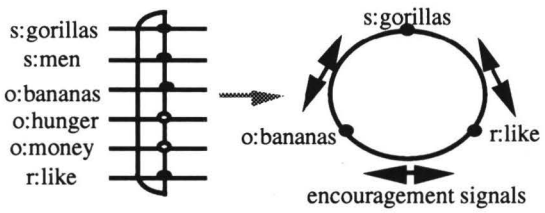

Figure 3 A Help-each-other Type Sub-network

Figure 4 A Winner-take-all Type Sub-network

3.3.2 Winner-take-all Type Sub-networks

The other type of sub-network, called the winner-take-all type network, is illustrated in Figure4. They prune the initial information units projected onto the winner predominant network. A sub-network chains all the nodes to which an identical information fragment is allocated. A live node tries to suppress the other live nodes within a sub-network.

3.3.3 Connection to/from the Hopfield Binary Network

The nodes of the winner predominant networks and the concept units of the Hopfield network are connected by bidirectional links. Activation signals are exchanged between the networks.

The activation signals from a winner predominant network on the left-hand (or right-hand) side are sent to the left (or right) bank of the Hopfield network just before every update operation term. This is to install an input activation pattern of Hopfield network units representing the information units held in the winner predominant network in the Hopfield network for the next update operation term. The update operation phase finally terminates after the signal flow in this direction so that the final information fragments left in the winner predominant networks will be handed over to the Hopfield network and sent out to the feedforward network.

On the other hand, the activation signals are sent out to the winner predominant network on the left-hand (or right-hand) side from the left (or right) bank of the Hopfield network just after an update operation term. This is to bring back the activation pattern of the Hopfield network obtained by the last update operation term to the winner predominant networks, and to check whether the information fragments represented by that pattern are still consistent with the information units held in the winner predominant network. The activation signal flow from the subject (or object) concept units to the winner predominant networks is only allowed, when no subject (or object) variable unit on the same bank is active. If a subject (or object) variable unit is active, then the activation signals from the concept units are ignored and the variable unit returns the activation signals to the winner predominant network. For this purpose, switching gates to control the activation signal flow are present on the links.

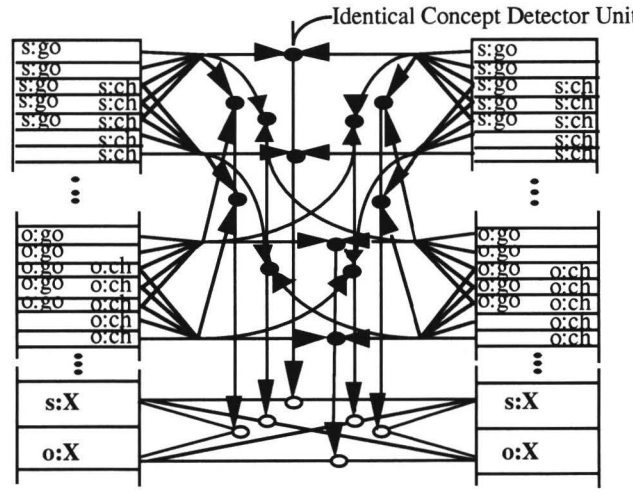

Figure 5 A Structural Constraint Network

3.4 The Structural Constraint Network

The structural constraint network is added to the system to provide the structural constraint imposed by the introduction of strict variables into the rules. The task imposed on the structural constraint network is to avoid inconsistent convergence of the Hopfield network to a unit activation pattern which involves inconsistent variable substitutions.

The network is divided into two sections. One is to check the existence of the semantically identical information fragments, and the other is to set the structural constraints between the semantically identical variables. It consists of the following components. See Figure5.

(1) Identical concept detector units each of which receives and compares the activation signals from two sets of concept units where a semantically, or both semantically and syntactically, identical concept is assigned.
(2) Unidirectional links between the concept units and the identical concept detector units.
(3) Weighted bidirectional links between pairs of variable units. In each pair, a semantically, or both semantically and syntactically, identical variable is represented.
(4) Unidirectional links between the identical concept detector units and the weighted bidirectional links.

If an identical concept detector unit detects that two sets of concept units are both almost active, it sends out a signal to the bidirectional links which connect the semantically corresponding pairs of variable units to adjust the weights of the links.

3.5 An Example Core Section

The Hopfield network, two winner predominant networks, and the structural constraint network form the core section of this system. Figure6 illustrates an example core section where the following collection of information fragments are implemented.

(1) Subject/object concepts:gorillas, chimps, fruits, bananas
(2) Subject/object variables: W, X, Y, Z
(3) Relations: like,eat,satisfy, belong, happy-with, dream

This system can hold rules consisting of these information fragments in the Hopfield network and feedforward network. For example, rule 1 as well as others like those presented below can be implemented in them.

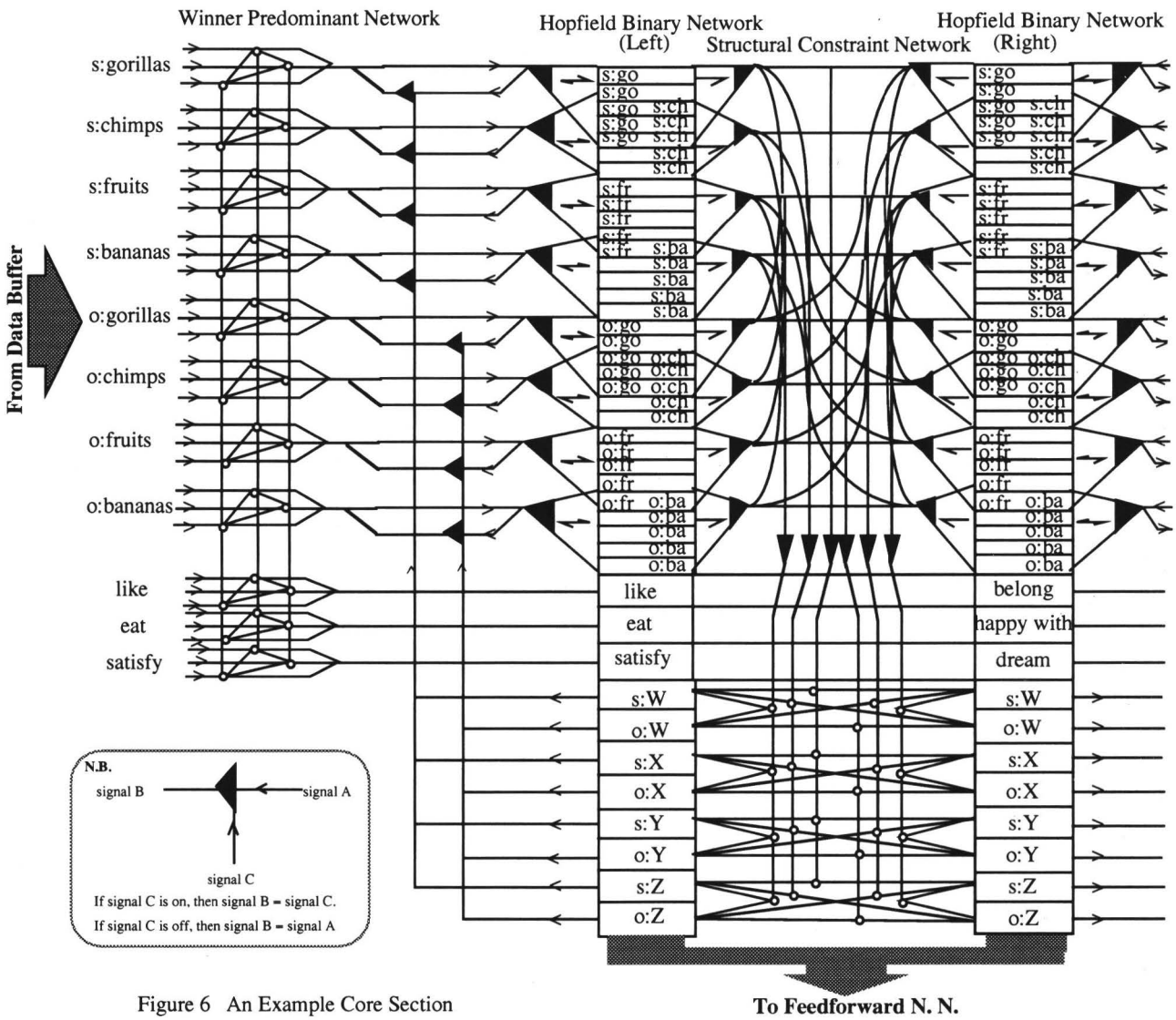

Figure 6 An Example Core Section

rule2: IF eat(s:X,o:Y) and happy-with(s:X,o:Y)
THEN dream(s:X,o:Y)

rule3: IF dream(s:X,o:Y) and eat(s:X,o:Y)
THEN like(s:X,o:Y)

3.6 The Feedforward Network

Once the Hopfield network has consistently converged to a rule premise pattern, the feedforward network takes charge of the derivation of a new information unit. It receives a collection of information fragments and variables as a pair of information units, and produces an information unit based on a rule implemented in the network. As the input activation patterns to the feedforward network may include probabilistic data owing to the similarity information implemented in the Hopfield network, the rule applications through the network can be probabilistic. For example, the network may receive

like with 100 % certainty, o:X,
s:chimps with 100 % certainty, s:gorillas with 50% certainty,
o:bananas with 100 % certainty, and o:fruits with 20 % certainty

from the left bank of the Hopfield network as the first premise of a rule and

belong with 100 % certainty, s:X,
s:bananas with 100% certainty, and s:fruits with 20 % certainty

from the right bank as the second premise, and produce

eat(s:gorillas,o:bananas) with 100 % or less (possibly 50 %) certainty

by rule1.

The network consists of three sections: input ports, output ports, and analogue units. Each input or output port is dedicated to representing one of the information fragments or variables represented in the Hopfield network. An input port receives a set of binary signals from the Hopfield network units where the same information fragment is assigned and converts them to an analogue activation signal to the analogue units. An output port displays an output activation signal for an information fragment and sends it to the data buffer.

Analogue units carry out the actual inferences. They are placed in three layers: input, hidden, and output layers. Each input or output unit is directly connected to an input or output port, whereas the hidden units are placed between them.

In order to provide the variable substitution mechanism to the system, extra circuits can be added onto the feedforward network so as to accomplish this task mechanically.

4 Network Training

4.1 The Hopfield Binary Network

Each of the two premises for a rule is stored in one of the two banks. For rule1, the first premise like(s:gorillas,o:X) is stored on the left bank of the Hopfield network as the activation pattern of units to which either like, s:gorillas or o:X is allocated, whereas the second premise belong(s:X,o:fruits) is stored on the right bank as the activation pattern of units to which either belong, s:X or o:fruits is allocated. The storage operation is completed by an automatic training procedure with the sum-of-outer-products function through which the appropriate weights are set on the connection arcs between Hopfield network units.

4.2 The Feedforward Network

The feedforward network is trained to implement the rule premise/conclusion relations by error back propagation. The premises of a rule are used as the input pattern for training, whereas the conclusion is used as the desired output pattern. For example, the input pattern to train rule1 is the activation signals to the input ports (of the feedforward network) for like, s:gorillas, and o:X, as well as belong, s:X, and o:fruits, and the desired output pattern is the activation signals to compare the output activation signals from the network at the output ports (of the feedforward network) for eat, s:gorillas and o:X. More discussion can be found in [Kozato, 93].

5 Update Operation Phase

5.1 The Winner Predominant Networks and The Hopfield Network

Since the initial input information to the Hopfield network includes several information units together and each information unit is merely a collection of information fragments, there is no way that the Hopfield network alone can correctly recognises the valid combinations of information fragments as autonomous information units during the update operation phase. Therefore, it is very probable that the network will converge to an activation pattern which represents no information unit stored in the data buffer but a strange combination of information fragments such as Figure7. In order to avoid such a situation, the Hopfield network needs to work in cooperation with the winner predominant networks. The cooperation task is performed after every update operation term of the Hopfield network by opening the connections between the concept and relation units of the Hopfield network and the nodes of the winner predominant networks.

If the state of the Hopfield network is changed after an update operation term so that some of the concept and relation units connecting to the live nodes of the winner predominant networks lower their activation level from the original level set by the live nodes, the activation level of those live nodes is also lowered. Accordingly, each such live node becomes less able to encourage the other live nodes within the same help-each-other type sub-network. This means that a certain information unit expressed by that help-each-other type sub-network tends to be removed from

the initial input information to leave out the redundant information for the rule selection.

On the other hand, every live node in a winner-take-all type sub-network tries to suppress the other live nodes according to the activation level. That is, the higher the activation level of a live node is, the more the node suppresses the others and gains its activation. Therefore, even if a live node receives less activation signals from the connected Hopfield network units, it can still regain its activation on condition that the other live nodes in the same help-each-other type sub-network have gained higher activation than the others belonging to the same winner-take-all type sub-network.

After this process, some of the information units in the winner predominant networks are weakened or weeded out. The information rearranged in the winner predominant networks is then reloaded to the Hopfield network for the next update operation term.

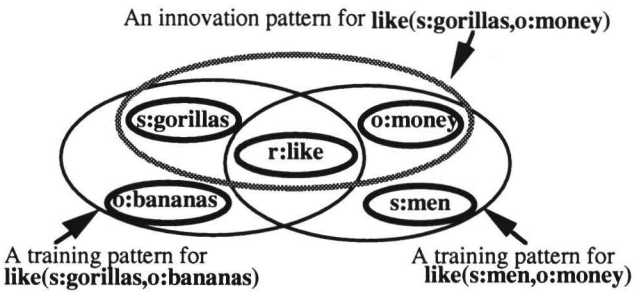

Figure 7 A Strange Information Unit

5.2 The Structural Constraint Network and The Hopfield Network

The Hopfield network also works in cooperation with the structural constraint network. This is to check whether there is any contradiction between the information units held in the winner predominant networks and the information represented by the unit activation pattern in the Hopfield network. Any premise pattern of a rule including variables should be expressed by the Hopfield network on condition that consistent variable substitution is ensured.

The units on both the banks of the Hopfield network representing a semantically, or both semantically and syntactically, identical concept are connected to an identical concept detector unit. Also, every pair of variable units representing a semantically, or both semantically and syntactically, identical variable on both the banks is connected by a bidirectional link with a certain weight.

In the beginning of every update operation term, each identical concept detector unit checks whether or not the two sets of concept units connecting to the identical concept detector unit are both active. If this is confirmed, the unit forces the negative weight of the corresponding links between variable units to be neutralised (±zero). For example (refer to Figure5), once the identical concept detector unit detects that the concept units representing s:gorillas on both the banks of the Hopfield network have been activated together, the identical concept detector unit sends out a signal to the weighted links between the variable units representing s:W, s:X, s:Y, and s:Z on both the banks so as to set the weights to zero. Accordingly, the variable units connected by the links are cut off so that these variable units are freed from the structural constraint.

6 Conclusion

Hopfield network, or even more generally, connectionist models, have not been designed to be capable of representing structured items in a simple manner nor processing each item discriminatory. To achieve a complicated task in such a system, some additional facilities for an extra mechanism or a large scale modification of the architecture to make it suitable for a particular use is necessary. In this model, the problem has been solved by the introduction of the structural constraint network for variable substitution and the winner predominant networks for pattern matching among several units of structured information.

References

[Kozato&De Wilde, 91a] Kozato, F. and De Wilde, P., "How Neural Networks Help Rule-based Problem Solvin ", proc. of the 1991 International Conference on Artificial Neural Networks, pp.465-470, Helsinki, 1991.

[Kozato&De Wilde, 91b] Kozato, F. and De Wilde, P., "A Probabilistic Rule-based System in Artificial Neural Networks", proc. of IEE Second International Conference on Artificial Neural Networks, pp.153-157, Bournmouth, U.K., 1991.

[Ballard, 86] Ballard, D. H., "Parallel Logical Inference and Energy Minimization", Technical Report no.142, Computer Science Department, University of Rochester, NY, March 1986. and also in proc. of AAAI, pp.203-208, 1986.

[Shastri, 88] Shastri, L., "A Connectionist Approach to Knowledge Representation and Limited Inference", Cognitive Science, vol.12, pp.311-392, 1988.

[Shastri & Ajjanagadde, 89] Shastri, L. and Ajjanagadde, V., "A Connectionist System for Rule Based Reasoning with Multi-Place Predicates and Variables", Technical Report, Computer and Information Science Department, University of Pennsylvania, Philadelphia, 1989.

[Hendler, 91] Hendler,J.A., "Developing Hybrid Symbolic/Connectionist Models" in High Level Connectionist Models, Barnden, J. A. and Pollack, J. B. (eds), vol.1, chap.7, Ablex Publishing Corporation, New Jersey, 1991.

[Touretzky& Hinton, 88] Touretzky, D.S. and Hinton, G.E., "A Distributed Connectionist Production System", Cognitive Science,vol.12,no.3, pp.423-466, 1988.

[Kozato, 93] Kozato, F., "Where Connectionist Models Meet Symbol Systems", PhD Thesis, Computing, Imperial College, University of London, London, 1993.