Symbol Generation and Integration from Neural Network Performing Reinforcement Learning

Y. Yamauchi & S. Tano
International Christian University, Division of Natural Science
3-10-2 Osawa, Mitaka, Tokyo, 1818585 Japan
ykr@icu.ac.jp
Graduate School of Information Systems, University of Electro-Communications
Chofu, Tokyo 182-8585, JAPAN
tano@is.uec.ac.jp

ABSTRACT

There are two methods in intelligent processing: computational (numerical information) processing and symbolic (knowledge-based) processing. Each of them has advantages and disadvantages. A simple model, integrating symbols to a neural network, had been proposed as the first step toward fusing computational and symbolic processing. In order to verify the proposed model, first, we analyze the trained neural network and generate symbols manually. Then we discuss generation methods that are able to discover effective symbols during the training of neural network. We have evaluated our methods by the simulations of reinforcement learning in the simple football games. The results indicate that the integration of symbols to the neural network improves the performance of the player agents.

I. INTRODUCTION

There are two types of intelligent processing: computational processing and symbolic processing (i.e., knowledge-based processing). Computational processing, such as fuzzy theory, neural network, and statistical method, appeared to have overcome the drawbacks of symbolic processing. These drawbacks include the difficulty in knowledge acquisition, the rigid inference and so on. However, the recent development of highly intelligent systems has exposed some serious limitations in computational processing. Two simple ways to overcome these limitations are to return to symbolism or to pursue a hierarchical architecture of systems. A hierarchical architecture of systems would combine both symbolic and computational processing. However, to achieve highly intelligent systems, these processing must be tightly interwoven in order to affect each other anytime, anywhere, and at any level of abstraction [1-13]. We overview our basic concept toward fusing computational and symbolic processing and propose a model to achieve our concept. Then we discuss some symbol generation and integration methods.

Our model consists of two layers. The lower layer is a neural network where the Q-learning is simulated. The inputs are the state variables, and the outputs are the Q-values for each action. It is tuned to update and store the Q-table. The upper layer watches the activity in the lower layer to identify the group of nodes that are activated when some action in the lower layer results in obtaining a high reward from the environment. In this way, new symbols emerge. These symbols are embedded in the lower layer to speed up the learning. When an important concept is learned, the corresponding symbol is generalized and embedded in a different place at a lower level.

In this paper, we will verify the proposed model using symbols that are generated based on analysis. We generate symbols manually based on two types of analysis. First, we analyze the activations and connections of a trained neural network. Second we analyze the role of input attributes and characteristic situations. Based on these analyses, symbols are generated and embedded manually into the neural network in early phase of training. We have evaluated these symbols by the simulations of reinforcement learning in the simple football games. The results indicate that the integration of symbols to the neural network improves the performance of the player agents. Finally we will propose and discuss generation methods to discover such effective symbols during the training of neural network automatically.

II. BASIC CONCEPT FOR DEEP FUSION

A. Requirements

Based on the analysis, we defined our goal to be a model towards the deep fusion of computational and symbolic processing that meets the following three requirements.

- The symbolic and computational processing should be tightly interwoven so as to utilize the advantages of each.
- Rather than simply be given *a priori* knowledge, the model should learn from the inputs, the outputs, and the rewards obtained from the environment.
- The learned knowledge should be generalized when possible, then used to improve performance.

B. Basic concept

To meet these requirements, we structured our model to incorporate the following concepts.

Computational Processing: The basic model is that of a neural network with back propagation. That is, computational processing is the fundamental processing mechanism.

Learning without a priori knowledge: To learn rather than

using *a priori* knowledge, we incorporated reinforcement learning, especially Q-learning.

Combined Q-learning and NN: The table for the Q-learning is represented by a neural network whose inputs are the attributes of the table and whose outputs correspond to the Q-value of the actions.

Symbolic Processing: As symbols gradually emerge from the NN, they are used to improve system inference performance and the learning ability.

C. Architecture

As shown in Fig. 1, our system is structured in two layers. The bottom layer is the Q-NN layer in which Q-learning occurs on the neural network. The system learns based on the results of the actions the system selects, the data it receives about the environment, and the rewards it obtains from the environment. The Q-table, which is updated based on this learning, resides in the Q-NN layer.

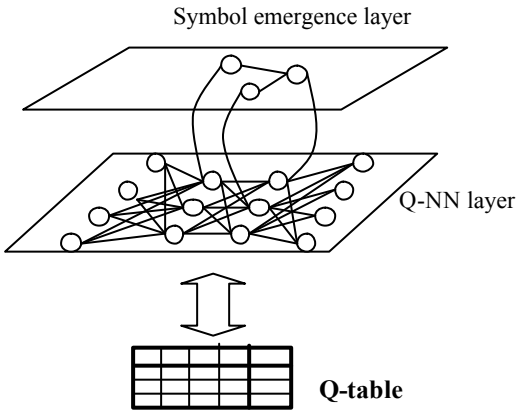

Fig. 1 System Architecture

The top layer is the symbol-emergence layer in which the activities in the Q-NN layer are constantly monitored. When an important concept is identified through this monitoring, it is embedded in the Q-NN layer as a node. The importance of a concept is judged based upon its relation with a good action. In other words, the system looks for concepts that lead to actions that bring high rewards.

Fig.2 shows the basic operation flow. The bottom layer is the neural network for Q-learning. In other words, the bottom layer updates and stores the Q-table. The system works by trying an action and checking the reward. Through this trial-and-error process, the system gradually obtains knowledge. This process is called "exploration". Although many trials are required, the system eventually learns a desirable series of action.

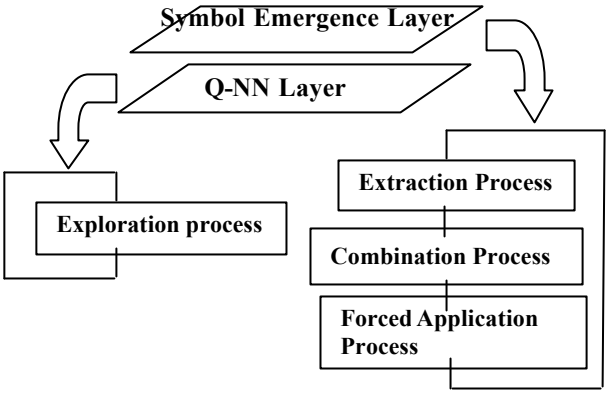

Fig. 2 Basic Processing Flow

The system extracts the important nodes by monitoring the Q-NN. In the top layer, the extraction process, the combination process and the forced application process are evoked sequentially. The extract process extracts an important concept. Its task is to find new concepts. The system then attempts to combine the extracted nodes to create an important concept in the combination process. The combined nodes can be regarded as describing an important concept.

III. SIMULATION SETTING

To evaluate the feasibility and effectiveness of our algorithm, we tested it by using a simulated football game. The size of the field was 13 x 9, and each team had two players. Each player can see its teammate and the two opposing players. When the game begins, the system does not have any knowledge. That is, a player does not even know that he gets a point when he kicks the ball into the opponent's goal. The players explore the game world by performing various actions in the various situations and getting rewards from the environment.

Each player is implemented by 2 feed forward neural networks, one for the situation for holding the ball (16 output nodes), and the other for not holding the ball (8 output nodes). Every neural network has 46 input nodes and 1 hidden layer with 20 hidden nodes.

The sample result of one simulation is shown in Table 1 and Fig. 3. A game is played between Q-learning team A and no learning (random move) team B. Table 1 shows the scores for each team per 20000 steps for early stage of simulation (first 200000 of 1000000). Fig. 3 shows the learning curve (measured by the score) of Team A implementing regular Q-learning compared to Team B. The score for team B becomes low as the players of team A learn their experience and improve to goal effectively.

Table 1 Scores for first 200000 steps

steps	A (Q-learn)	B (random)
20000	0	12
40000	2	11
60000	1	4
80000	1	6
100000	2	7
120000	5	5
140000	19	6
160000	43	5
180000	49	3
200000	95	2

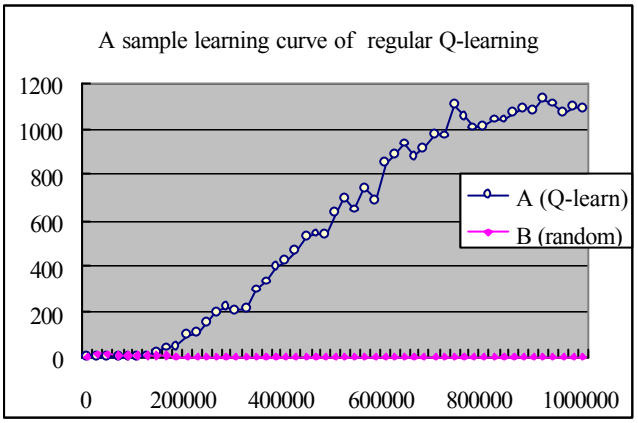

Fig. 3 Learning Curve of the Q-learning

IV. SYMBOL CONSTRUCTION BASED ON ANALYSIS

In this section we will verify the proposed model using symbols that are constructed based on analysis. First, we analyze the trained neural network to determine important nodes and links and construct symbols combining them. Second, we analyze the input attributes and construct symbols to represent meaningful situations for certain actions. These symbols are embedded into the original neural network in early phase of training.

A. Analysis of Trained Neural Network

We analyze activation circumstances of neural network and intensity of an association between each node and build a symbol on the basis of node that gives important influence.

First, the number of strong activation when positive reward is acquired is counted for each node of the hidden layer. For example, Fig. 4 shows the statistics of the activations of the nodes of hidden layer for Agent0 when it does not hold the ball, and nodes 16, 5, 8, 18 are frequently activated compared to others.

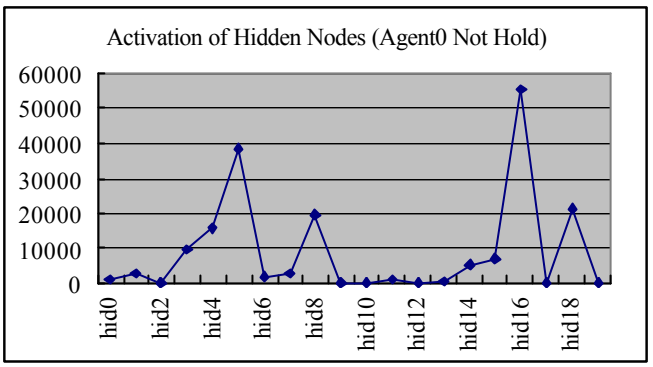

Fig. 4 Activation of Hidden Nodes

Then we analyze the association between frequently activated hidden nodes and all input nodes. Fig. 5 shows the Input-Hidden layer association (link = weight) of Agent0 not holding the ball. As shown in Fig. 5, hidden node 16 is strongly associated with input node 31, 33 and 28 that indicates the direction of the ball.

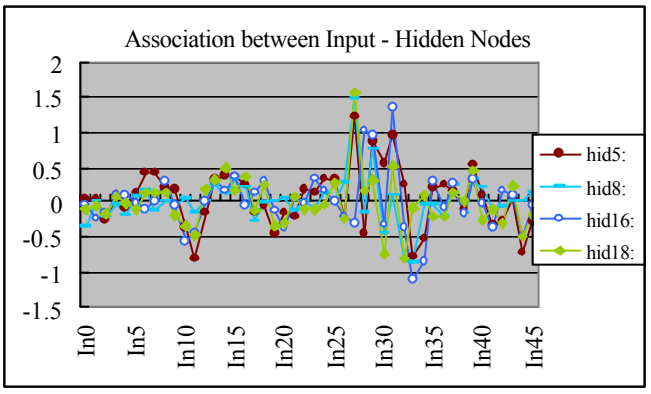

Fig. 5 Association of Input-Hidden Nodes

Finally we analyze the association between frequently activated hidden nodes and all output nodes. Then we combine hidden nodes that are strongly associated to the same output nodes and connect the strongly associated input nodes. Constructed symbols with fixed links are embedded into the neural network at early phase of training and connected to the all output nodes with small weight. These symbol-output associations are learned when neural network is trained after integration as shown in Fig. 6. The fixed part is the key of our model that represent a concrete (symbolic) concept.

We have tested 100 trials of Q-learning (Team A) against Random (Team B) and 100 trials of Q-learning with Symbols (Team SN) against Random (Team B). 5 symbols are integrated at 300000 steps. Fig. 7 shows the average learning curve of Q-learning with symbol integration (SN) compared to the performance of Q-learning only (A). Team S gain more score than Team A and gave less score to Team B than Team A did.

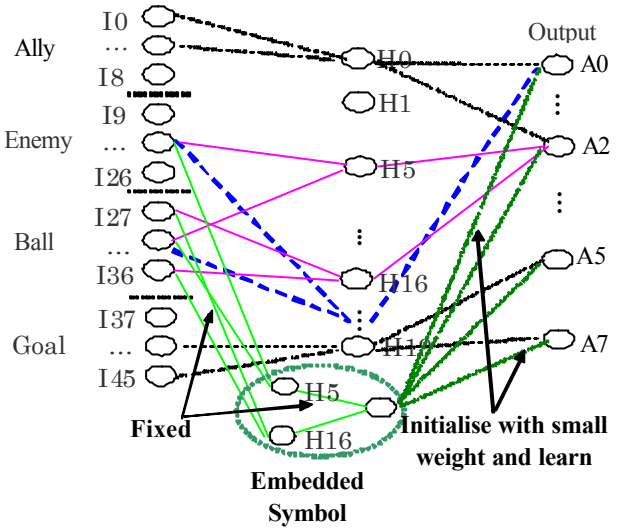

Fig. 6 Integration of Constructed Symbol

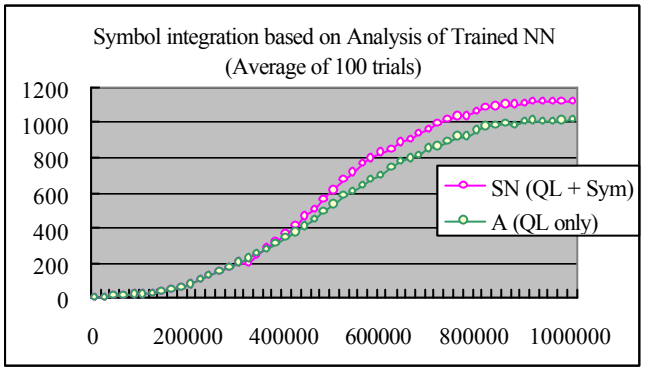

Fig. 7 Symbol Integration (NN Analysis)

Simulation demonstrated that integration of symbols based on the analysis of trained neural network improves the performance of reinforcement learning.

B. Analysis of Input Attributes and Situations

We have found that the symbols constructed by the analysis of trained neural network are effective to integrate the proposed model in previous discussion. Thus, we analyze the role of these symbols and consider input attributes to construct symbols that represent meaningful situations for certain actions.

When a player does not have the control of the ball (NOT HOLD), the position of the ball is the most important concept to consider. However if the ally player has the ball or is closer to the ball, he should consider other action rather than going straight to the ball (such that moving near the goal to get a pass from his ally). Based on this analysis, the input attributes for the position of ball and ally are combined to construct a symbol that represent a situation "when the ball is in X direction and the ally is NOT that direction" as shown in Fig. 8.

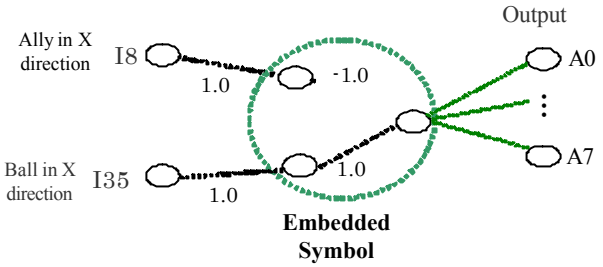

Fig. 8 Symbol Represents Ally and Ball in same Direction (Not Hold)

Similarly, when a player has the control of the ball (HOLD), the position of the goal is the most important input to consider. However if the enemy player is closer to the goal, he should try to pass the ball to his ally or dribble away from the enemy. Based on this analysis, the input attributes for the position of goal and enemies are combined to construct a symbol that represent a situation "when the goal is in X direction and the enemies are NOT in the same direction" as shown in Fig. 9.

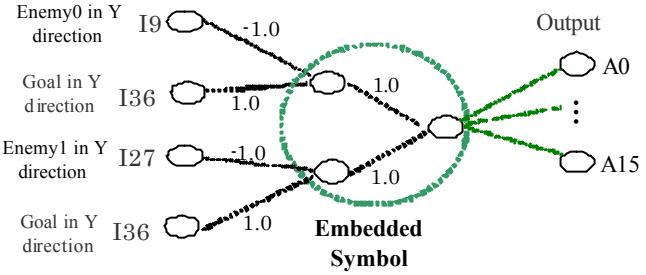

Fig. 9 Symbol Represents Goal and Enemies in same Direction (Hold)

We have tested 100 trials of Q-learning with Symbols based on analysis of input attributes (Team SA) against Random (Team B). Fig. 10 shows the average learning curve of Q-learning with symbol integration (SA) compared to the performance of Q-learning only (A). Team SA gain more score than Team A and gave less score to Team B than Team A did.

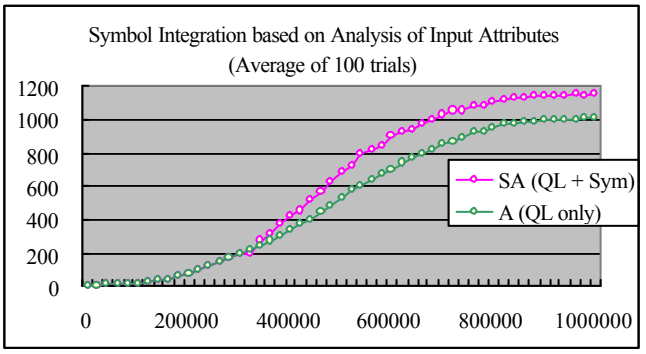

Fig. 10 Symbol Integration (Input Attributes Analysis)

Simulation demonstrated that integration of symbols based on the analysis of trained neural network and input attributes both improved the performance of reinforcement learning without symbols.

V. SYMBOL GENERATION

Embedded symbols generated by the analysis improved the performance of Q-learning neural networks and verified effectiveness of our model fusing computational and symbolic processing. What become important is how to generate these symbols during the training of neural network by an algorithm. We introduce a simple generation methods similar to the process of analysis on trained neural network.

The extraction process, the integration process and the forced application process work as follows.

(i) Extraction Process

Step1: Assume that system received a reward when it executed action A.

Step2: Store list of activated nodes that affected action node A. For example. H2 and H3 in Fig. 11.

Step3: Store list of activated nodes that affect the nodes stored in Step 2. For example, H2: I2, I5 and H3: I6, I9.

Step4: Repeat Steps 2 and 3 until sufficient data have been collected.

Step5: Identify nodes that are often activated when action A is executed.

(ii) Integration Process

Step1: Check output nodes for extracted hidden nodes.

H1 (I1, I4) … A3
H2 (I2, I5) … A3
H3 (I6, I9) … A1

Step2: Judge which nodes are connected to the same output that should represent similar situation.

Step3: Unify the similar nodes into one concept by creating a new node and connecting it to the nodes selected in *Step2*.

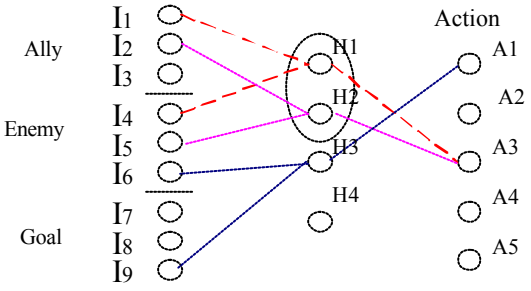

Fig. 12 Combination of Nodes

(iii) Forced Application Process

Step1: Assume that when the system identifies an important concept, it combines the related nodes into a new node and embeds the new node. H1 and H2 are new nodes representing important concepts in the example shown in Fig. 12.

Step2: New nodes in network. N1, N2, and S1 are embedded in the network shown in Fig. 13.

Step3: Connect new node to all nodes in the next layer at a low weight. In Fig. 13, S1 is connected to a1 to a5.

We have tested 100 trials of Q-learning with symbols generated by the proposed algorithm. Extraction process starts at 150000 steps, and the maximum number of symbols to be generated is limited to 10 for a trial of 1000000 simulation steps. Fig. 14 shows the remarkably better performance when symbols are extracted and integrated to the Q-learning neural network.

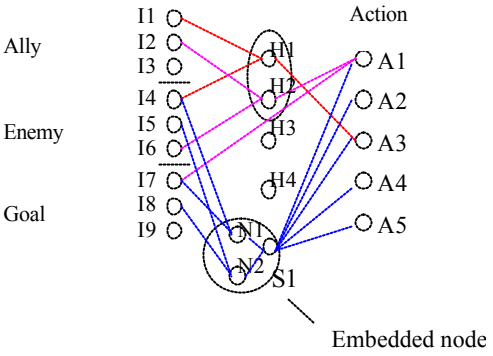

Fig. 13 Forced Application of Symbol

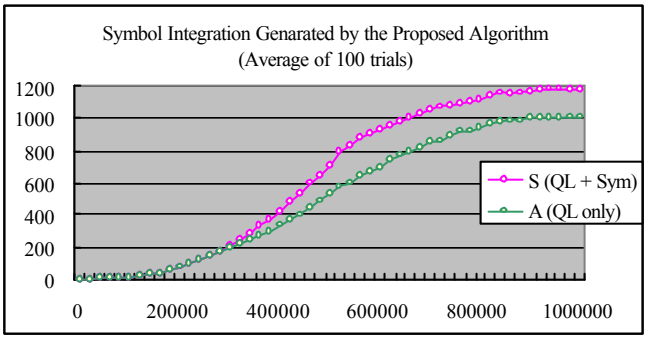

Fig. 14 Symbol Integration (Generated by the Algorithm)

VI. CONCLUSION

We believe the key for the highly intelligent system is to enable interaction between computational and symbolic processing anytime, anywhere, and at any level of abstraction. By *synergistically* combining symbols and patterns, powerful mechanisms should emerge. We call this approach "*Deep Fusion of Computational and Symbolic Processing*".

We have proposed a simple model as the first step toward fusing computational and symbolic processing and verified our model integrating symbols constructed on the basis of analysis. Then, we proposed an algorithm that extract symbols from neural network, integrate the important concepts and embed into neural network and perform integrated learning.

Though the idea is quite simple, our model differs from other symbol extraction methods from trained neural network such as [15] in following points. First, we extract and generate symbols *during* the training periods. Second, we extract symbols and use them in a *unified model* (in the neural network itself). This feature is the major difference from other hybrid systems. Also, since we fix the conections (weights) of

our symbols as the concrete concepts, our model shows the clear difference from other models that optimize or dynamically change the structure of neural network represented in [16].

Through simulations we demonstrated that symbol emergence and the forced application of these symbols in Q-learning greatly improves the performance of neural network applied to the agents playing a simple football game.

REFERENCES

[1] R. Sun and T. Peterson, "Hybrid Learning Incorporating Neural and Symbolic Processes", FUZZ-IEEE'98 (1998)

[2] T. Omori and N. Yamanashi, "PATON A Model of Concept Representation and Operation in Brain", Proc. of Int'l Vonfon Neural Network 94, pp. 2227-2232 (1994)

[3] I. Takeuchi and T. Furuhashi: "A Study on Inference between Patterns and Symbols", 13th Fuzzy System Symposium, pp. 573-576 (1997) (in Japanese)

[4] T. Takagi, A. Imura, H. Ushida, and T. Yamaguchi, "Computational Fuzzy Sets as a Meaning Representation and Their Inductive Construction", International Journal of Intelligent Systems, Vol.10, No. 11, pp. 929-945 (November 1995)

[5] S. Ohsuga, "Symbol Processing by Non-Symbol Processor", Proc.4th Pacific Rim International Conference on Artificial Intelligence, Cairns, Australia (1996)

[6] S. Tano, "Synergetic Effect by Deep Fusion of Computational and Symbolic Processing", FUZZ-IEEE'98 (1998)

[7] Y. Uemura and S. Tano: "Analysis of symbolic and computational processing and a new fusion method", 14th Fuzzy System Symposium, pp. 179-180 (1998) (in Japanese)

[8] Y. Uemura , D. Futamura, and S. Tano: "Proposal of Symbolic and Computational Processing and Initial Evaluation by Simulation", 15th Fuzzy System Symposium, pp. 471-474 (1999) (in Japanese)

[9] Papers in the organized session "Deep Fusion of Computational and Symbolic Processing" of FUZZ-IEEE' 98 at World Congress on Computational Intelligence (WCCI' 98)

[10] S. Tano, Miyoshi, Kato, Oyama, Arnould, and Bastian: "Fuzzy Inference Software - FINEST: Overview and Application Examples", IEEE International Conference on Fuzzy Systems, FUZZ-IEEE'95, pp. 1051-1056, (1995)

[11] S.Tano, Oyama, and Arnould: "Deep Combination of Fuzzy Inference and Neural Network in Fuzzy Inference Software - FINEST", International Journal of Fuzzy Set and Systems, (1996)

[12] S. Tano, Okamoto, and Iwatani: "New Design Concepts for the FLINS-Fuzzy Lingual System: Text-based and Fuzzy-centered Architectures", International Symposium on Methodologies for Intelligent Systems – ISMIS'93, pp. 285-294, (1993)

[13] S. Tano, et al.: "Three-layered Fuzzy Inference and Self-wondering Mechanism as Natural Language Processing Engine of FLINS", IEEE International Conference on Tools with Artificial Intelligence – TAI'94, pp. 212-218, (1994)

[14] S. Tano, et al.: "Symbol Emergence and its Forced Application in Q-learning toward Fusion of Computational and Symbolic Processing", IEEE-SMC '99, vol. III, pp. 827-832, (1999)

[15] Z. Zhou, Y. Jiang and S. Chen: "Extracting Symbolic Rules from Trained Neural Network Ensembles", AI Communications 2003 16(1), pp. 3-15, (2003)

[16] G Towell and J. Shavlik: "Knowledge-Based Artificial Neural Networks", Artificial Intelligence, 70(1-2), pp. 119-165, (1994)