Abstraction of Continuous System to Discrete Event System Using Neural Network

Sung Hoon Jung* and Tag Gon Kim†,

*School of Information and Computer Engineering, Hansung Univ., Seoul, Korea
†Department of Electrical Engineering, KAIST, Taejon 305-701, Korea

ABSTRACT

A hybrid system consists of continuous systems and discrete event systems, which interact with each other. In such configuration, a continuous system can't directly communicate with a discrete event system. Therefore, a form of interface between two systems is required for possible communication. An interface from a continuous system to a discrete event system requires abstraction of a continuous system as a discrete event system. This paper proposes a methodology for abstraction of a continuous system as a discrete event system using neural network. A continuous system is first represented by a timed state transition model and then the model is mapped into a neural network by learning capability of the network. With a simple example, this paper describes the abstraction process in detail and discusses application methods of the neural network model. Finally, an application of such abstraction in design of intelligent control is discussed.

Keywords: Models Abstraction, Discrete Event Model, Neural Network

1. INTRODUCTION

Recently, intelligent control schemes for discrete event systems have been extensively researched.[1-4] In such control schemes, a discrete event system is at the heart of an autonomous control system.[5-8] To control a continuous system using the intelligent control schemes, the continuous system should be first abstracted to an higher level system. Especially when a system is a complex hybrid one, composed of continuous and discrete event systems, the continuous system should be abstracted to a discrete event system for possible communication between the controller and the continuous systems. This is because direct communication between them is impossible due to the different level of information for each system. An interface from a continuous system to a discrete event system requires abstraction of a continuous system as a discrete event system.

This paper proposes a methodology for abstraction of a continuous system as a discrete event system using neural network. Within the methodology, a discrete event system is represented by a timed state transition model. Thus, abstraction is to find state transition rules along with delay times between states after defining a finite state set for the continuous system. Discrete elements of a timed state transition model can be obtained from operational objectives of a continuous system through three steps — output partitioning, input sampling, and measuring of delay time . This timed state transition model is mapped to a neural network using a learning algorithm. More specifically, we first define a finite state set for the continuous system to be abstracted. Next, a neural network is trained by using a set of state transitions and associated delay times between states. In this paper, we present the model abstraction process for a continuous system represented in a set of differential equations.

This paper is organized as follows. Section 2describes the brief review of a discrete event system specification and neural network. In section 3.the abstraction process from a continuous system to a discrete event system is explained with an example system. Application of the neural network model to intelligent control is discussed in section 4. . Section 5concludes this paper.

For further information -
*S.H.J.(correspondence): Email: shjung@ice.hansung.ac.kr; Telephone: +82-2-760-4344; Fax: +82-2-760-4217
†T.G.K.: Email: tkim@ee.kaist.ac.kr; Telephone: +82-42-869-3454; Fax: +82-42-869-3410

SPIE Vol. 3083 • 0277-786X/97/$10.00

2. BRIEF REVIEW OF BASIC METHODOLOGIES

2.1. Discrete event system specification formalism

A discrete event system can be modeled by a discrete event system specification (DEVS) formalism proposed by Zeigler.[9,10] The DEVS formalism is as follows.

$$M = < X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta > \quad (1)$$

where

- X is the input events set
- S is the sequential state set
- Y is the output events set
- $\delta_{int} : S \rightarrow S$ is the internal transition function
- $\delta_{ext} : Q \times X \rightarrow S$ is the external transition function where $Q = \{(s, e) \mid s \in S, 0 \leq e \leq ta(s)\}$
- $\lambda : S \rightarrow Y$ is the output function
- $ta : S \rightarrow R^{+}_{0,\infty}$ is the time advanced function

The input/output event set describes all inputs/outputs of a system, and the state set includes all possible states of the system. The state transition when external events are received is appeared on the external transition function. The state transition, when the time specified by the time advanced function is elapsed, is presented by the internal transition function. When the internal transition is occurred, the output which is expressed by the output function is also generated. More detailed informations about the formalism are available in.[9,10] Based on this formalism, we will introduce a timed state transition formalism for abstracted continuous systems in section 3.

2.2. Artificial neural network

Artificial Neural Networks (ANN) consist of highly interconnected simple processing elements called *neurons*. Each neuron consists of a summing junction, which adds together the weighted inputs from the other neurons, and an activation function, which generates the neuron output from the summing junction output. The output fans out to serve as an input to other neurons. Neurons transmit signals to each other via weighted links, which attenuate or amplify the transmitted signal depending on the weight value. The advantages of neural networks are twofold: learning capability and versatile mapping capabilities from inputs to outputs. Learning is a mechanism for storing knowledge about the external world, and for acquiring skills and knowledge of how to act.

A back-propagation neural network (BPNN) as a kind of artificial neural network is formalized by McClelland and Rummelhart.[11] The structure of BPNN is a fully interconnection between the neighboring layers, no feedback connections between layers and no interconnections among neurons in the same layer. There is a typical standard back-propagation algorithm for training the back-propagation neural network. The operation of the algorithm consists of two phases: propagate and adapt phases. In propagate phase, errors for input data are computed, then the weight values of the network are updated by a gradient descent method in the adapt phase. The total sum square error (TSSE) for all output neurons is reduced by the updating weights of the neural network. The learning (or training) process minimizes the total sum square error given as;

$$E = \frac{1}{2}\sum_{p=1}^{R} E^p = \frac{1}{2}\sum_{p=1}^{R}\sum_{i=1}^{K}(t_i^p - o_i^p)^2 \quad (2)$$

where R is the number of data in the training set, K is the number of output neuron, and E^p is the error value for pattern p. t_i^p and o_i^p are the target and actual output of pattern p in output neuron i, respectively.

Back-propagation neural networks have been used in many application areas such as associative mapping, classification problem, system identification, and so on. The associative mapping is to describe the relation of mapping among data, and the classification is to categorize input data for some classes. In system identification, features of a system are extracted according to various objectives.

3. ABSTRACTION METHODOLOGY

This section describes the abstraction process, and an abstraction example with a simple continuous system.

3.1. Abstraction process

Figure 1 shows the abstraction processes. The abstraction processes are as follows. When a continuous system is

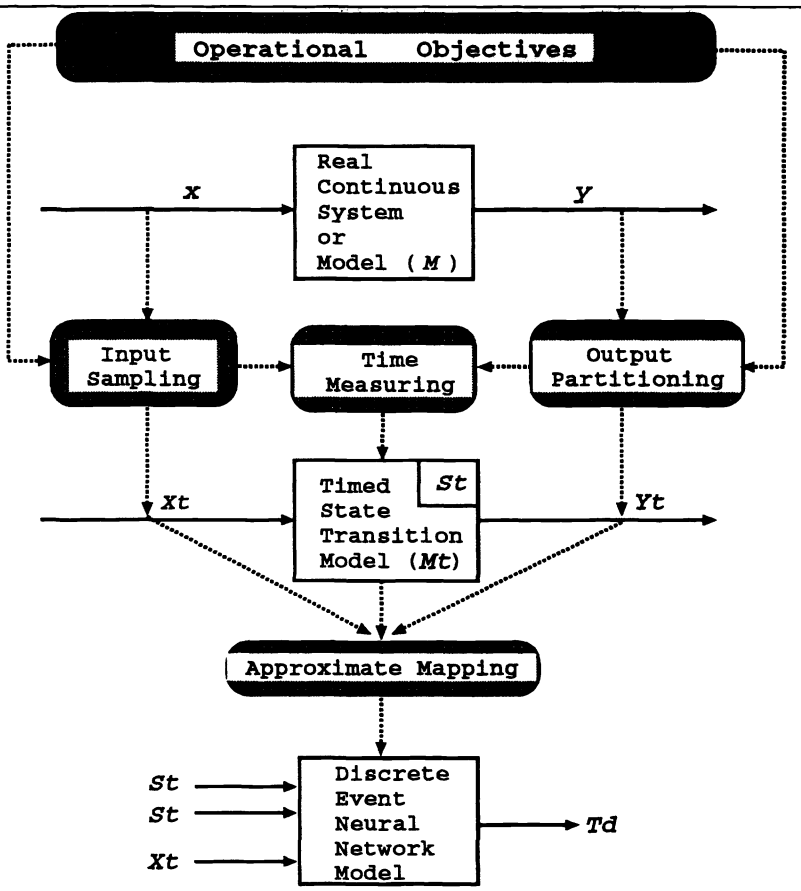

Figure 1. Abstraction Process of Continuous System

given, a system designer must first decide the operational objectives of the continuous system. The operational objectives can be derived from functional requirements and constraints of the system. To get the discrete inputs and outputs of the continuous system, the designer must perform output partitioning followed by input sampling. The output partitioning is to divide the outputs of the continuous system into several mutually exclusive blocks to quantize the output levels considering operational objectives of the system. The input sampling is to select some inputs for state transitions specified by the operational objectives . Third, delay times for the state transitions under given inputs must be measured using a real continuous system or a model such as a set of differential equations. With these informations, we can construct a timed state transition model for the continuous system.

Based on the DEVS formalism in section 2.1.we can specify a timed state transition formalism for continuous systems with a 6-tuple as follows.

$$M = (X, Y, S, T_d, \lambda, s_0) \tag{3}$$

where

- X is the input events set obtained from input sampling
- Y is the output events set obtained from output partitioning
- S is the states set
- $Td : S \times X \times S \rightarrow R^{+}_{0,\infty}$ is the time delay function
- $\lambda : S \rightarrow Y$ is the output function
- s_0 is the initial state in S

In this formalism, the state set is the same as the output events and an output event always generates the state information of the continuous system. Note that the time delay function implicitly includes the state transition function. Those abstracted informations such as states and state transition times can be managed by many methods. A table model that represents such informations as a tabular form can be a method. However, the management of the table model is very tedious and difficult when operational objectives are dynamically changed. To dynamically manage the abstracted informations, we used the learning capability of back-propagation neural network. For mapping a timed state transition model to a neural network, the inputs and outputs of the neural network should be defined. The input and output events sets in equation 3 cannot be assigned for outputs of the neural network. This is because the neural network will generate outputs different from those of the training data owing to incomplete training. This makes it impossible to decide which event of the input and output events set are matched to the outputs of the neural networks. However, the delay time of state transitions can be assigned for the output of the neural network because the time is able to have any real values in $R^{+}_{0,\infty}$ as shown in equation 3. From this observation, we decide the inputs and outputs of neural network. The inputs of neural network are composed of a current state (in other words, a current output) value, a target state (in other words, a target output) value, and an input value. With these three inputs, the neural network generates the delay time for the state transition, i.e., from a current state to a target state as shown in figure 1. Of course, all values must be in predefined each set as elements.

3.2. An abstraction example

With a simple water-tank example, we illustrate the abstraction process. Figure 2 shows a water-tank continuous system. Let the constraints of the water tank system and operational objectives of this example be given as follows, respectively.

- Constraints
  - The water tank must be able to supply the water to another device with two rates (not zeros).
  - The water can be supplied from another device with two rates (not zeros)
- Operational Objectives
  - The water level must be kept in the vicinity of the middle of the water-tank not to go over the top of the water-tank and not to go under the bottom of the water-tank.
  - Time to rise the water level from the bottom level to the middle level must not be over 2 minutes.
  - Time to fall the water level from the top level to the middle level must not be over 3 minutes.

From the first operational objective, we can first do output partitioning with three levels, i.e., $High - Mark, Midd - Mark, Low - Mark$. Of course, three threshold sensors for detecting the three levels must be equipped to the water-tank for real control. Using the second and third operational objectives and two constraints, we can next do input sampling. For simplicity, four values are selected for the input valve and output valve, $High - Input, Low - Input, High - Output, Low - Output$. Of course, those symbolic elements for each set is assigned for real value, for example, $High - Mark = 3m$, $High - Input = 10lb/min$ and so on. Finally, delay times for state transitions under given inputs should be measured using a real continuous system or a differential equations model.

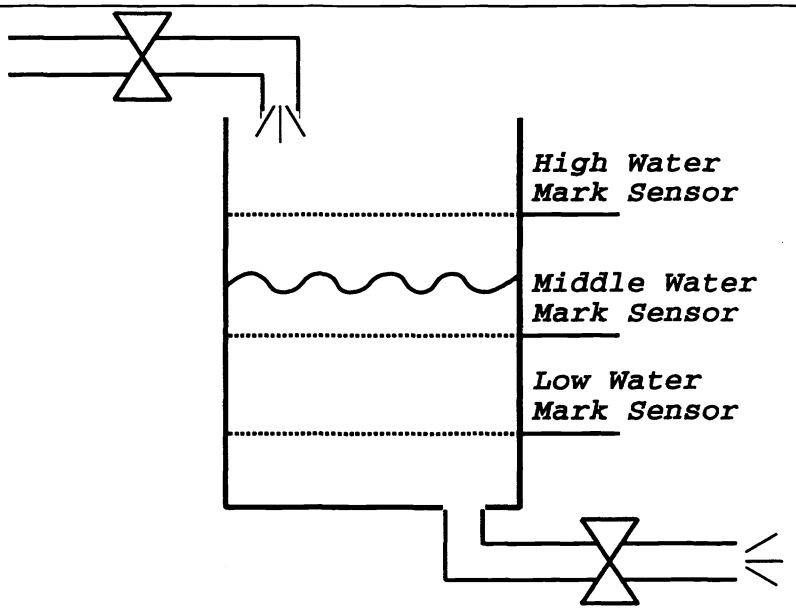

Figure 2. Water Tank Continuous System

The water-tank operation is simply modeled using a first-order differential equation as follows.

$$C\frac{dh}{dt} = q_i - q_o \tag{4}$$

where C is the capacity of water-tank, h is the height of water-tank, q_i is the input flow rate of water, and q_o is the output flow rate of water. The height of water-tank is obtained by solving the equation 4.

$$h(t) = h_0 + \frac{q_i - q_o}{C}t \tag{5}$$

From this differential equation, the delay time can be calculated by following equation.

$$Td = \frac{h_c - h_0}{\frac{q_i - q_o}{C}} \tag{6}$$

where Td is delay time, h_c is a target state, and h_0 is a current state. Of course, the elements of h_c, h_0, q_i, q_o must be in the predefined set.

The dynamics of a real water tank will not be exactly the same as those of the differential equation model. This is caused not only by modelling errors but also by parameter changes of the real water tank and its environment. Let the minimum and maximum parameter variation be P_{min}, P_{max} respectively, then the minimum and maximum times to reach a specific height h_c are as follows.

$$\begin{aligned} t_{min} &= \frac{h_c - h_0}{(\frac{q_i - q_o}{C})p_{max}} \\ t_{max} &= \frac{h_c - h_0}{(\frac{q_i - q_o}{C})p_{min}} \end{aligned} \tag{7}$$

Consequently, we can get a time window by taking these minimum and maximum times:

$$t_{min} \leq t_{win} \leq t_{max} = \frac{C(h_c - h_0)}{(q_i - q_o)P_{max}} \leq t_{win} \leq \frac{C(h_c - h_0)}{(q_i - q_o)P_{min}}$$

Let the C, P_{min}, and P_{max} be 12.2, 0.9037, and 1.12963 respectively, then the time window is given as:

$$\frac{10.8(h_c - h_o)}{(q_i - q_o)} \leq t_{win} \leq \frac{13.5(h_c - h_o)}{(q_i - q_o)} \tag{8}$$

This time window can be used for diagnosis when the system is controlled by event-based intelligent control or by supervisory control as will be described in section 4. These gathered data is mapped to the neural network with a back-propagation algorithm.

Let the output states (level of the water) and inputs (flow rates of input and output valves) be partitioned and sampled as shown in table 1, respectively. To satisfy the first operational objective and two constraints, the high input rate of water to fill the water tank must be greater than the high output rate not to be empty. Also, the low input rate of water to sink the water must be lower than the low output rate not to be overflowed. We can measure

Table 1. Real Values of States and Inputs of Water Tank

Water Mark			Two Input			
HIGH_TANK	MID_TANK	LOW_TANK	HIGH_IN	LOW_IN	HIGH_OUT	LOW_OUT
22.5	15.0	3.5	12.5	2.5	8.5	4.5

the minimum and maximum times for each state transition under given inputs. Table 2 shows the gathered data from the equation 8. In this table, the minus time means that the delay time is infinity, that is, the target state will never be reached under given inputs.

The gathered data is mapped to a neural network with 4-20-15-2 network structure. We used 0.25 learning rate and 10000 iterations. After learning, state trajectories under given inputs are checked with the differential equation model and neural network model. For the state trajectories, we assume that no parameter changes of the water tank and environment exists. Then the simulated differential equation is the same as the equation 5. Figure 3 shows the state trajectories. In figure 3, we selected such inputs not to make the delay times infinite for drawing. The x-axis means simulation time under given inputs as shown in table 3. In any current state, of course, delay time for any target state can be retrieved from the neural network model.

4. APPLICATION TO INTELLIGENT CONTROL

This section introduces the application of a neural network model to the event-based intelligent control and supervisory control. A neural network model for a continuous system can be used as an internal model in event-based intelligent control. In event-based intelligent control, the controller has an internal model whose dynamics is the same as that of a controlled system at discrete event levels. With this internal model, the controller can do diagnosis the controlled system. That is, if a state transition from a current state to a target state occurs within the time constraints (i.e., minimum time and maximum time for the state transition), then the controller regards the current control operation as being correct. Otherwise, an error is assumed. If continuous systems are a part of a hybrid systems with discrete event systems, then internal neural network models for continuous systems are used as shown in figure 4. With these neural network models, the event-based intelligent control works even if the controlled system is a hybrid system. More detailed descriptions about the event-based intelligent control are available in.[3,4]

In supervisory control, the water tank model abstracted can be defined as a 5-tuple

$$M = (Q, \Sigma, \delta, q_0, Q_m) \tag{9}$$

where

- Q is the state set, $Q = \{Empty, LOW\_TANK, MID\_TANK, HIGH\_TANK, Overflow\}$
- Σ is the events set, Σ = {*HIGH_IN, LOW_IN, HIGH_OUT,LOW_OUT, LOW_TANK_IND, MID_TANK_IND, HIGH_TANK_IND*}

Table 2. Gathered Data From Differential Equation Model of Water Tank

Water Mark		Two Input		Time Window	
Current State	Target State	Input Rate	Output Rate	Minimum Time	Maximum Time
LOW_TANK	MID_TANK	HIGH_IN	HIGH_OUT	31.1	38.8
LOW_TANK	MID_TANK	HIGH_IN	LOW_OUT	15.5	19.4
LOW_TANK	MID_TANK	LOW_IN	HIGH_OUT	-20.7	-25.9
LOW_TANK	MID_TANK	LOW_IN	LOW_OUT	-62.1	-77.6
LOW_TANK	HIGH_TANK	HIGH_IN	HIGH_OUT	51.3	64.1
LOW_TANK	HIGH_TANK	HIGH_IN	LOW_OUT	25.7	32.1
LOW_TANK	HIGH_TANK	LOW_IN	HIGH_OUT	-34.2	-42.8
LOW_TANK	HIGH_TANK	LOW_IN	LOW_OUT	-102.6	-128.2
MID_TANK	LOW_TANK	HIGH_IN	HIGH_OUT	-31.1	-38.8
MID_TANK	LOW_TANK	HIGH_IN	LOW_OUT	-15.5	-19.4
MID_TANK	LOW_TANK	LOW_IN	HIGH_OUT	20.7	25.9
MID_TANK	LOW_TANK	LOW_IN	LOW_OUT	62.1	77.6
MID_TANK	HIGH_TANK	HIGH_IN	HIGH_OUT	20.2	25.3
MID_TANK	HIGH_TANK	HIGH_IN	LOW_OUT	10.1	12.7
MID_TANK	HIGH_TANK	LOW_IN	HIGH_OUT	-13.5	-16.9
MID_TANK	HIGH_TANK	LOW_IN	LOW_OUT	-40.5	-50.6
HIGH_TANK	LOW_TANK	HIGH_IN	HIGH_OUT	-51.3	-64.1
HIGH_TANK	LOW_TANK	HIGH_IN	LOW_OUT	-25.7	-32.1
HIGH_TANK	LOW_TANK	LOW_IN	HIGH_OUT	34.2	42.8
HIGH_TANK	LOW_TANK	LOW_IN	LOW_OUT	102.6	128.2
HIGH_TANK	MID_TANK	HIGH_IN	HIGH_OUT	-20.2	-25.3
HIGH_TANK	MID_TANK	HIGH_IN	LOW_OUT	-10.1	-12.7
HIGH_TANK	MID_TANK	LOW_IN	HIGH_OUT	13.5	16.9
HIGH_TANK	MID_TANK	LOW_IN	LOW_OUT	40.5	50.6

Table 3. Inputs for State Trajectory

X-axis		State Transition		Inputs	
From	To	CS	TS	Input Rate	Output Rate
0	35.1	LOW_TANK	MID_TANK	HIGH_IN	HIGH_OUT
35.2	46.5	MID_TANK	HIGH_TANK	HIGH_IN	LOW_OUT
46.6	162.4	HIGH_TANK	LOW_TANK	LOW_IN	LOW_OUT
162.5	220.3	LOW_TANK	HIGH_TANK	HIGH_IN	HIGH_OUT
220.4	235.5	HIGH_TANK	MID_TANK	LOW_IN	HIGH_OUT
235.6	305.6	MID_TANK	LOW_TANK	LOW_IN	LOW_OUT
305.7	323.1	LOW_TANK	MID_TANK	HIGH_IN	LOW_OUT
323.2	346.5	MID_TANK	LOW_TANK	LOW_IN	HIGH_OUT
346.6	375.5	LOW_TANK	HIGH_TANK	HIGH_IN	LOW_OUT
375.6	421.3	HIGH_TANK	MID_TANK	LOW_IN	LOW_OUT
421.4	444.2	MID_TANK	HIGH_TANK	HIGH_IN	HIGH_OUT
444.3	482.8	HIGH_TANK	LOW_TANK	LOW_IN	HIGH_OUT

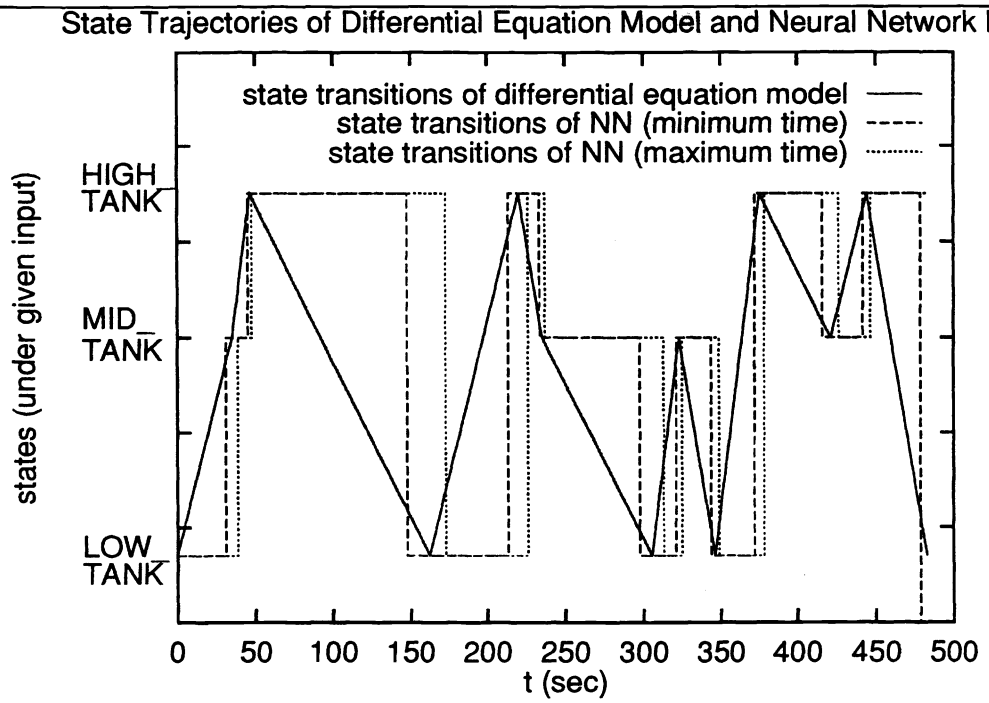

Figure 3. State Trajectories of Differential Equation Model and Neural Network Model

- $\delta : Q \times \Sigma \times \Sigma \rightarrow Q$ is the transition functions, for examples,
  - $\{LOW\_TANK, (HIGH\_IN, HIGH\_OUT)\} \rightarrow MID\_TANK$
  - $\{LOW\_TANK, (LOW\_IN, HIGH\_OUT)\} \rightarrow Empty$
  - $\{HIGH\_TANK, (LOW\_IN, HIGH\_OUT)\} \rightarrow MID\_TANK$
  - $\{HIGH\_TANK, (HIGH\_IN, HIGH\_OUT)\} \rightarrow Overflow$
  - $\{HIGH\_TANK, (HIGH\_IN, LOW\_OUT)\} \rightarrow Overflow$
- $q_0 \in Q$ is the initial state, $q_0 := LOW\_TANK$
- $Q_m \subset Q$ is maker states, $Q_M := MID\_TANK$

The events, $LOW\_TANK\_IND, MID\_TANK\_IND$, and $HIGH\_TANK\_IND$, are indicative events of three states, respectively. In this model, let the controllable events and uncontrollable events be given as:

$$\begin{aligned} \Sigma &= \Sigma_c \cup \Sigma_{uc} \\ &= \{HIGH\_IN, LOW\_IN, HIGH\_OUT, LOW\_OUT\} \cup \\ &\quad \{LOW\_TANK\_IND, MID\_TANK\_IND, HIGH\_TANK\_IND\} \end{aligned} \tag{10}$$

where Σ_c and Σ_{uc} are controllable and uncontrollable events, respectively. Then, a supervisor can drive from any states to the maker state because we select that any composition of flow rates of inputs and outputs can fill or sink the water.

In timed supervisory control, abstracted neural network model can also be used for diagnose. For example, when a supervisory controller enables two events, $LOW\_IN, HIGH\_OUT$ and disables two events, $HIGH\_OUT$, LOW_OUT, for state transition from $HIGH\_TANK$ to MID_TANK, the event MID_TANK_IND should occur within a time

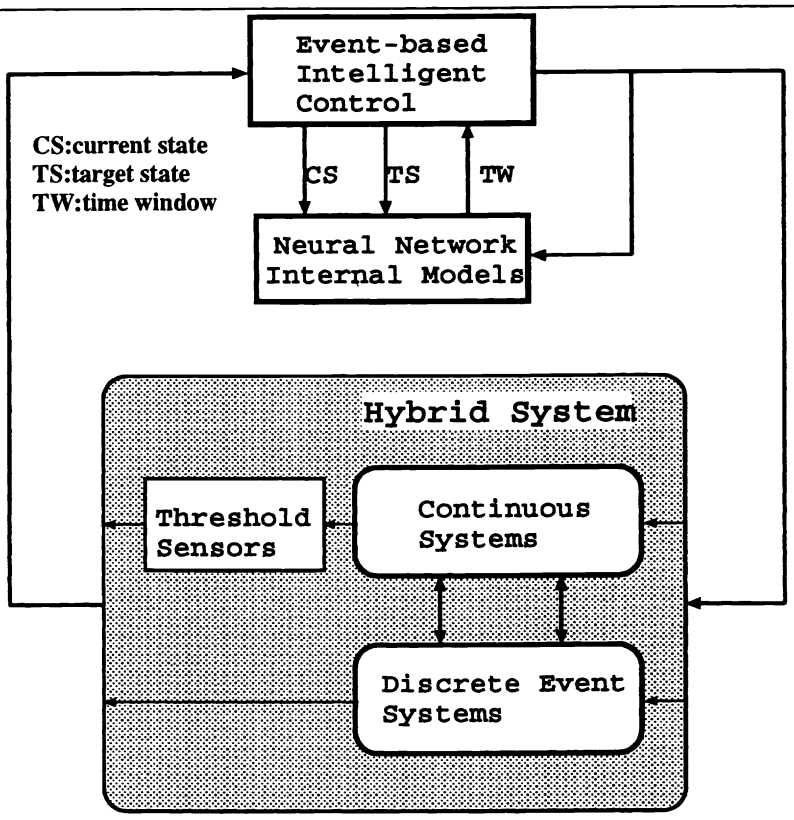

Figure 4. Event-based Intelligent Control of A Hybrid System

window generated from the neural network model. In all applications of abstracted neural network model, learning capability of neural network makes it possible to dynamic modelling of controlled system by on-line learning. This scheme is very similar to human control strategy in that human can learn more and more informations about the controlled system as the control actions proceed.

5. CONCLUSION

This paper introduced the abstraction process of a continuous system to a discrete event system. With operational objectives, we could first do two processes, output partitioning and input sampling. Next, a timed state transition model for a continuous system was constructed by measuring delay time between states transition. For on-line learning of the timed state transition model, we used a neural network. We discussed an application of the neural network model in intelligent control. More research for incorporating a supervisory control and abstracted neural network model for diagnosis is further work.

REFERENCES

1. P. J. Ramadge and W. M. Wonham, "Supervisoty control of a class of discrete event processes," *SIAM J. Control and Optimization*, vol. 25, pp. 206–230, Jan. 1987.
2. J. R. Clymer, P. D. Corey, and J. A. Gardner, "Discrete Event Airport Control," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 22, pp. 343–351, MARCH/APRIL 1992.
3. B. P. Zeigler, "DEVS Representation of Dynamical Systems: Event-Based Intelligent Control," *Proceedings of the IEEE*, vol. 77, pp. 72–80, Jan. 1989.

4. B. P. Zeigler and J. Kim, "Extending The DEVS-Scheme Knowledge-Based Simulation Environment for Real-Time Event-Based Control," *IEEE Trans. on Robotics and Automation*, vol. 9, no. 3, pp. 351–356, 1993.
5. C.-J. Luh and B. P. Zeigler, "Abstracting Event-Based Control Models for High Autonomy Systems," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 23, pp. 42–54, JANUARY/FEBRUARY 1993.
6. A. P.J., K. Passino, and S. Wang, "Towards Intelligent Autonomous Control Systems: Architecture and Fundamental Issues," *Journal of Intelligent and Robotic Systems*, vol. 1, no. 4, pp. 315–342, 1989.
7. S. H. Jung, T. G. Kim, and K. H. Park, "Event-Based Intelligent Control Using Endomorphic Neural Network Model," *An International Journal Applied Artificial Intelligence*, vol. 9, pp. 479–494, September-October 1995.
8. S. H. Jung, T. G. Kim, and K. H. Park, "Event-Based Intelligent Control of Saturated Chemical Plant Using Endomorphic Neural Network Model," *Journal of Intelligent Manufacturing*, vol. 6, pp. 365–376, Dec. 1995.
9. B. P. Zeigler, *Theory of Modelling and Simulation.* John Wiley, 1976.
10. B. P. Zeigler, *Multi-Faceted Modelling and Discrete Event Simulation.* Academic Press, 1984.
11. McClelland and D. Rummelhart, *Parallel Distributed Processing*, vol. 1 and 2, MIT Press, Cambridge, MA, 1986.