[11] W. L. Goffe, G. D. Ferrier, and J. Rogers, "Global optimization of statistical functions with simulated annealing," *J. Econometrics*, vol. 60, pp. 65–99, 1994.
[12] C. R. Houck, J. A. Joines, and M. G. Kay, "A genetic algorithm for function optimization: A Matlab implementation," submitted for publication.
[13] A. Torn and A. Zillinskas, *Global Optimization*. Berlin, Germany: Springer-Verlag, 1987.
[14] R. Horst and P. M. Pardalos, *Handbooks of Global Optimization*. Amsterdam, The Netherlands: Kluwer, 1995.

Temporal Knowledge Representation and Reasoning Techniques Using Time Petri Nets

Woei-Tzy Jong, Yuh-Shin Shiau, Yih-Jen Horng, Hsin-Horng Chen, and Shyi-Ming Chen

***Abstract*—In this paper, we present temporal knowledge representation and reasoning techniques using time Petri nets. A method is also proposed to check the consistency of the temporal knowledge. The proposed method can overcome the drawback of the one presented in [16]. It provides a useful way to check the consistency of the temporal knowledge.**

***Index Terms*—Knowledge representation, rule-based system, temporal knowledge, time Petri nets.**

I. Introduction

The concept of time plays a very important role in our lives. In order to solve the temporal knowledge representation and reasoning problem, developing a system that can store and manipulate the knowledge about time is necessary. In [1], Allen described 13 kinds of relations of time, where each of the 13 relations represents the order of two time intervals. In [16], Yao pointed out that there are mainly two kinds of representation and reasoning schemes for temporal information, i.e., Dechter's linear inequalities [6] to encode metric relations between time points and Allen's temporal calculus [1]. Each scheme has its advantages and disadvantages. In [12], Kautz *et al*. introduced a model to integrate two schemes for temporal reasoning in order to benefit from the advantages of each scheme. In [8], Dutta presented an event-based fuzzy temporal logic. It can determine effectively the various temporal relations between uncertain events or their combinations. In [7], Deng *et al*. presented a G-Net for knowledge representation and reasoning. In [5], we presented a fuzzy Petri net model (FPN) to represent the fuzzy production rules of rule-based systems and presented a fuzzy reasoning algorithm to deal with fuzzy reasoning in rule-based systems. However, the models presented in [5] and [7] cannot be used for temporal knowledge representation. In [16], Yao presented a model based on time Petri nets for handling both qualitative and quantitative temporal information. In [4], we pointed out that the method presented in [16] has a drawback in checking the consistency of temporal knowledge.

Manuscript received February 6, 1998; revised January 30, 1999. This work was supported in part by the National Science Council, R.O.C., under Grant NSC 86-2213-E-009-018.

W.-T. Jong, Y.-S. Shiau, Y.-J. Horng, and H.-H Chen are with the Department of Computer and Information Science, National Chiao Tung University, Hsinchu, Taiwan, R.O.C.

S.-M. Chen is with the Department of Electronic Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan, R.O.C.

Publisher Item Identifier S 1083-4419(99)05278-4.

In this paper, we present a method to describe the relationships between states and events using time Petri nets for temporal knowledge representation and reasoning. We also present an algorithm to check the consistency of temporal knowledge. The proposed method can overcome the drawback of the one presented in [16].

The rest of the paper is organized as follows. In Section II, we introduce the basic concepts and definitions of time Petri nets. The temporal knowledge representation techniques using time Petri nets are also presented in Section II. In Section III, we present some operations between time intervals and between paths in a time Petri net. In Section IV, we present an algorithm to check the consistency of temporal knowledge. The conclusions are provided in Section V.

II. Time Petri Nets

In this section, we introduce the basic concepts of time Petri nets. A time Petri net is a bipartite directed graph which contains two types of nodes, i.e., places and transitions, where circles represent places and bars represent transitions. There are several definitions of time Petri nets [11], [16]. A time Petri net is a ten-tuple $(S, E, P, T, B, F, M_0, \alpha, \beta,$ SIM), where

S	finite set of states, $S = \{S_1, S_2, \cdots, S_n\}$;
E	finite set of events, $E = \{E_1, E_2, \cdots, E_m\}$, where each event is associated with a transition;
P	finite set of places, $P = \{P_1, P_2, \cdots, P_n\}$, where each place is associated with a state;
T	finite set of transitions, $T = \{t_1, t_2, \cdots, t_m\}$, where each transition is associated with a time interval;
B	backward incidence function, B: $T \times P \rightarrow N$, where N is the set of nonnegative integers;
F	forward incidence function, F: $T \times P \rightarrow N$;
M_0	initial marking function M_0: $P \rightarrow N$;
α	mapping function from places to states, α: $P \rightarrow S$;
β	mapping function from events to transitions, β: $E \rightarrow T$;
SIM	mapping function called static interval mapping function, SIM: $T \rightarrow Q^*$, where Q^* is a time interval.

In a time Petri net, each transition is associated with a time interval $[a, b]$, where a is called the static Earliest Firing Time, b is called the static Latest Firing Time, and $a \leq b$, where

1) a $(0 \leq a)$ is the minimal time that must elapse, starting from the time at which the transition is enabled until the transition can fire;
2) b $(0 \leq b \leq \infty)$ represents the maximum time during which the transition is enabled without being fired.

The values of a and b are relative to the moment the transition is enabled. If the transition is enabled at time λ, then the transition cannot be fired before time $\lambda + a$, and the transition must be fired before time $\lambda + b$.

In a time Petri net, a place may contain tokens. A time Petri net with some places containing tokens is called a marked time Petri net. For example, Fig. 1 shows a marked time Petri net, where events E_1, E_2, E_3, and E_4 are associated with time intervals, $[t_{11}, t_{12}], [t_{21}, t_{22}], [t_{31}, t_{32}]$, and $[t_{41}, t_{42}]$, respectively. An arc from a place to a transition defines the place to be the input (backward incidence) place of the transition. An arc from a transition to a place defines the place to be the output (forward incidence) place of the transition. A transition is enabled if and only if each of its input places has a token. When a transition is enabled, it may be fired. When a transition fires, all tokens are removed from its input places,

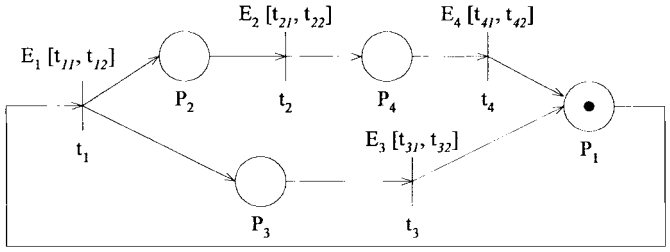

Fig. 1. Marked time Petri net.

and a token is added into each of its output places. For example, in Fig. 1, transition t_1 is enabled because there is a token in place P_1 (P_1 is the only input place of t_1). After t_1 is fired, the token in P_1 is removed and each of the places P_2 and P_3 has a token. In a marked time Petri net, the places initially containing tokens are called initial marking places.

In [1], Allen describes thirteen possible relationships between two time intervals. Yao [16] modeled these relationships using time Petri nets. Assume that place P_i is associated with state S_i (i.e., $\alpha(P_i) = S_i$), where $1 \leq i \leq n$, event E_j is associated with transition t_j (i.e., $\beta(E_j) = t_j$), where $1 \leq j \leq m$, and transition t_j is associated with the time interval $[j_1, j_2]$. In [10], we have used time Petri nets for temporal knowledge representation.

III. Operations Between Time Intervals and Between Paths

In this section, we present the operations between time intervals and between paths in a time Petri net [10].

Definition 3.1: Let P_k be a place, and let t_i and t_j be transitions in a time Petri net. If P_k is a forward incidence place of t_i and P_k is a backward incidence place of t_j, then we say that the forward incidence place of t_i coincides with the backward incidence place of t_j.

Definition 3.2: Assume that the time interval $T_1 = [a, b]$, where $0 \leq a \leq b \leq \infty$, and time interval $T_2 = [c, d]$, where $0 \leq c \leq d \leq \infty$. Then

1) Time Interval Union ($\cup$):
   - Case 1: If $b < c$, then $T_1 \cup T_2 = [a, b] \cup [c, d]$.
   - Case 2: If $a \leq c \leq b \leq d$, then $T_1 \cup T_2 = [a, d]$.
   - Case 3: If $c \leq a \leq d \leq b$, then $T_1 \cup T_2 = [c, b]$.
   - Case 4: If $d < a$, then $T_1 \cup T_2 = [c, d] \cup [a, b]$.
2) Time Interval Intersection ($\cap$):
   - Case 1: If $b < c$, then $T_1 \cap T_2 = \phi$.
   - Case 2: If $a \leq c \leq b \leq d$, then $T_1 \cap T_2 = [c, b]$.
   - Case 3: If $c \leq a \leq d \leq b$, then $T_1 \cap T_2 = [a, d]$.
   - Case 4: If $d < a$, then $T_1 \cap T_2 = \phi$.
3) Time Interval Addition ($+$): $T_1 + T_2 = [a + c, b + d]$.

Definition 3.3: Two time intervals T_1 and T_2 are *joint* if $T_1 \cap T_2$ is not empty (i.e., $T_1 \cap T_2 \neq \phi$).

Definition 3.4: In a time Petri net, a *path* is a sequence of transitions $\{t_i, t_{i+1}, \cdots, t_j\}$ such that the output place of t_k coincides with the input place of t_{k+1} for $i \leq k \leq j - 1$, where the path is a set of transitions.

Definition 3.5: Two transitions t_i and t_j are *contradictory* if t_i and t_j are enabled by the same backward incidence places.

Definition 3.6: Let path_1 and path_2 be two paths in a time Petri net, where $\text{path}_1 = \{t_g, t_{g+1}, \cdots, t_a, \cdots, t_h\}$ and $\text{path}_2 = \{t_i, t_{i+1}, \cdots, t_b, \cdots, t_j\}$. If t_a and t_b are not contradictory, then the union of the two paths (i.e., $\text{path}_1 \boxdot \text{path}_2$) is equal to $\{t_g, t_{g+1}, \cdots, t_a, \cdots, t_h\} \cup \{t_i, t_{i+1}, \cdots, t_b, \cdots, t_j\}$, where $\cup$ is the union operator of sets. If t_a and t_b are contradictory, then we let the result of $\text{path}_1 \boxdot \text{path}_2$ be ϕ.

Let t_i be a transition and let $\{t_j, t_{j+1}, \cdots, t_k\}$ be a path in a time Petri net. Adding t_i into the path is expressed as $\{t_j, t_{j+1}, \cdots, t_k\} \bigcirc + t_i = \{t_j, t_{j+1}, \cdots, t_k, t_i\}$.

Let path_b be a path and let PS be a set of paths, where $PS = \{\text{path}_1, \text{path}_2, \cdots, \text{path}_a\}$. Adding path_b into PS is defined as $PS \boxplus \text{path}_b = \{\text{path}_1, \text{path}_2, \cdots, \text{path}_a, \text{path}_b\}$.

Definition 3.7: Let PS_1 and PS_2 be two sets of paths, where $PS_1 = \{\text{path}_g, \text{path}_{g+1}, \cdots, \text{path}_h\}$ and $PS_2 = \{\text{path}_i, \text{path}_{i+1}, \cdots, \text{path}_j\}$. The *Cartesian union* operation between PS_1 and PS_2 is defined by

$$\begin{aligned} PS_1 \diamondsuit PS_2 &= \{\text{path}_g \boxdot \text{path}_i, \text{path}_g \boxdot \text{path}_{i+1}, \cdots \\ &\quad \text{path}_g \boxdot \text{path}_j, \text{path}_{g+1} \boxdot \text{path}_i \\ &\quad \text{path}_{g+1} \boxdot \text{path}_{i+1}, \cdots, \text{path}_{g+1} \boxdot \text{path}_j \\ &\qquad \vdots \\ &\quad \text{path}_h \boxdot \text{path}_i, \text{path}_h \boxdot \text{path}_{i+1} \cdots, \\ &\quad \text{path}_h \boxdot \text{path}_j\} \end{aligned}$$

where $\boxdot$ is the Union operator between paths. The *merge* operation between PS_1 and PS_2 is defined by

$$PS_1 \diamondsuit PS_2 = \{\text{path}_g, \text{path}_{g+1}, \cdots, \text{path}_h, \text{path}_i, \text{path}_{i+1}, \cdots, \text{path}_j\}.$$

Adding a transition t_a into PS_1 is defined as $PS_1 \boxdot t_a$, where

$$PS_1 \boxdot t_a = \{\text{path}_g \oplus t_a, \text{path}_{g+1} \oplus t_a, \cdots, \text{path}_h \oplus t_a\}.$$

Definition 3.8: Let TS_1 and TS_2 be two sets of time intervals, where $TS_1 = \{T_g, T_{g+1}, \cdots, T_h\}, TS_2 = \{T_i, T_{i+1}, \cdots, T_j\}$, and let T_a be some time interval. The union of TS_1 and TS_2 is defined as $TS_1 \cup TS_2$.

$$TS_1 \cup TS_2 = \{T_g, T_{g+1}, \cdots, T_h, T_i, T_{i+1}, \cdots, T_j\}.$$

The *multiplication* of TS_1 and TS_2 is defined by

$$\begin{aligned} TS_1 \otimes TS_2 = \{&T_g \cap T_i, T_g \cap T_{i+1}, \cdots, T_g \cap T_j \\ &T_{g+1} \cap T_i, T_{g+1} \cap T_{i+1}, \cdots, T_{g+1} \cap T_j \\ &\qquad \vdots \\ &T_h \cap T_i, T_h \cap T_{i+1}, \cdots, T_h \cap T_j\} \end{aligned}$$

where $\cap$ is the intersection operator of time intervals. Adding a time interval T_a into a set TS_1 of time intervals, is defined as $TS_1 \boxdot T_a$ where

$$TS_1 \boxdot T_a = \{T_g \cup T_a, T_{g+1} \cup T_a, \cdots, T_h \cup T_a\}$$

where $\cup$ is the union operator of time intervals.

IV. Temporal Reasoning Using Time Petri Nets

In this section, we present an algorithm for performing temporal reasoning using time Petri nets. The algorithm essentially constructs a sprouting graph, where each node is associated with an ordered pair (a, b), where a indicates the current place and b is a triplet, and each directed arc (includes dashed directed arc) is associated with an ordered pair (c, d), where c indicates the current transition and d is the time interval associated with the transition in the time Petri net. The triplet of the ordered pair of a node in the sprouting graph consists of a time interval, a set of paths, and a set of time intervals, where the time interval represents the possible time of occurrence of the current

place of the node, each path in the set of paths consists of transitions passed through from the initial marking places to the current place of the node, and each time interval in the set of time intervals represents the time that the corresponding path needs to spend. If the ith path in the set of paths is ϕ, then the ith time interval of the set of time intervals is ϕ as well. The algorithm constructs a sprouting graph to model the transfer of tokens in the marked time Petri net. It uses the sprouting graph to check the consistency of temporal knowledge and to perform temporal reasoning. Assume that there are n places and m transitions in a time Petri net. The algorithm consists of two steps.

Step 1) Generate a sequence of transitions of the marked time Petri net. This step can be divided into the following substeps.

1) Let F_1 be a $n \times m$ backward incidence matrix. If the place P_i is the backward incidence place of the transition t_j, then set $F_1(P_i, t_j) = 1$. Otherwise, set $F_1(P_i, t_j) = 0$.
2) Let F_2 be a $n \times m$ forward incidence matrix. If the place P_i is the forward incidence place of the transition t_j, then set $F_2(P_i, t_j) = 1$. Otherwise, set $F_2(P_i, t_j) = 0$. However, if the place P_i is the initial marking place of the marked time Petri net, then set $F_2(P_i, t_j) = 0$ for every transition t_j of the marked time Petri net.
3) Find a place P_i that has never been found such that $F_2(P_i, t_j) = 0$ for every transition t_j of the marked time Petri net, and set $F_1(P_i, t_j) = 0$ for every transition t_j of the marked time Petri net.
4) Find a transition t_j that has never been found such that $F_1(P_i, t_j) = 0$ for every place P_i of the marked time Petri net, then output the transition t_j, set $F_2(P_i, t_j) = 0$ for every place P_i of the marked time Petri net, and go to (3). If we can't find any transition t_j such that $F_1(P_i, t_j) = 0$ for every place P_i of the marked time Petri net or we have already output all transitions, then go to Step 2.

Step 2) Construct the sprouting graph. This step can be divided into the following substeps.

1) Create a node for every initial marking place of the marked time Petri net, where the first value of the ordered pair associated with the node is this initial marking place and the triplet of the ordered pair associated with this node is $([0, 0], \phi, \phi)$ unless the user defines it. These nodes are called root nodes.
2) If the firing sequence generated in Step 1 is $t_1 t_2 \cdots t_k$, then select the first transition of the firing sequence, i.e., let $t_j = t_1$. Assume that the time interval associated with t_j is T_a.
   i. Find the nodes in which the first value of the ordered pair associated with each node is one of t_j's backward incidence places. Assume that the triplet of the ordered pairs of these nodes are $(I_1, \text{Path\_set}_1, I\_\text{Path\_set}_1)$, $(I_2, \text{Path\_set}_2, I\_\text{Path\_set}_2)$, $\cdots$, $(I_r, \text{Path\_set}_r, I\_\text{Path\_set}_r)$, where I_i is a time interval, Path_set_i is a set of paths, and $I\_\text{Path\_set}_i$ is a set of time intervals, $1 \le i \le r$. Let

$$I = (I_1 \cap I_2 \cap \cdots \cap I_r) + T_a,$$

$$\text{Path\_set} = (\text{Pat\_set}_1 \diamondsuit \text{Path\_set}_2 \diamondsuit \cdots \diamondsuit \text{Path\_set}_r) \boxdot t_j,$$

$$I\_\text{Path\_set} = (I\_\text{Path\_set}_1 \otimes I\_\text{Path\_set}_2 \otimes \cdots \otimes I\_\text{Path\_set}_r) \boxdot T_a.$$

   If $(I_1 \cap I_2 \cap \cdots I_r) = \phi$ (i.e., the time intervals are not joint), then let $j = j + 1$ and go to iv). Otherwise, t_j is called firable.
   ii. Find the forward incidence places of t_j. Assume that these places are $P_1, P_2, \cdots, P_s$. For each $P_i, 1 \le i \le s$.

   Case 1: If P_i is the initial marking place of the marked time Petri net, then draw a dash directed arc from each node we find in i) to the node in which the first value of the ordered pair associated with this node is P_i and let the ordered pair associated with the dash directed arc be (t_j, T_a).

   Case 2: If there exists a node in which the first value of the ordered pair associated with this node is P_i and assume that the triplet of the ordered pair associated with the node is $(I_a, \text{Path\_set}_a, I\_\text{Path\_set}_a)$, then draw a directed arc which is associated with the ordered pair (t_j, T_a) from each node we find in i) to the node and set the triplet of the ordered pair associated with the node to be $(I \cup I_a, \text{Path\_set} \not\diamondsuit \text{Path\_set}_a, I\_\text{Path\_set} \cup I\_\text{Path\_set}_a)$.

   Case 3: If we can't find any node in which the first value of the ordered pair associated with this node is P_i, then create a new node, and let the ordered pair associated with the created node be $(P_i, (I, \text{Path\_set}, I\_\text{Path\_set}))$, and draw a directed arc which is associated with the ordered pair (t_j, T_a) from each node we find in i) to the created node.
   iii. Let $j = j + 1$.
   iv. If $j > k$, then terminate and the sprouting graph has been created, otherwise, go to i).

Assume that the ordered pair associated with a node of the sprouting graph is $(P_i, (I_i, \text{Path\_set}_i, I\_\text{Path\_set}_i))$, where I_i is a time interval, Path_set_i is a set of paths, and $I\_\text{Path\_set}_i$ is a set of time intervals. Furthermore, assume that $I\_\text{Path\_set}_i = \{T_{i1}, T_{i2}, \cdots, T_{ik}\}$, where k is a positive integer and P_i is not an initial marking place, then we can find that $I_i = T_{i1} \cup T_{i2} \cup \cdots \cup T_{ik}$.

In the following, we present a method to check the consistency of the temporal knowledge and to perform temporal reasoning. If there exists a place that doesn't appear in any ordered pair associated with the node of the graph, then we say that this marked time Petri net is not consistent [16]. In other words, if we define some temporal knowledge that won't happen in any case, then the corresponding marked time Petri net would be not consistent.

However, how can we know that it is possible that the transition t_i and the transition t_j of the marked time Petri net occur at the same time? Furthermore, if we know that t_i and t_j are possible to occur at the same time, then what events (or transitions) should occur before? Assume that the time interval associated with t_i is $[i_1, i_2]$, the backward incidence places of t_i are $P_{i1}, P_{i2}, \cdots, P_{ia}$, and assume that the triplet associated with the node of which the first value of the ordered pair is P_{ix} is $(I_{ix}, \text{Path\_set}_{ix}, I\_\text{Path\_set}_{ix})$, where $1 \le x \le a$. The time interval associated with t_j is $[j_1, j_2]$, the backward incidence places of t_j are $P_{j1}, P_{j2}, \cdots, P_{jb}$, and assume that the triplet associated with the node of which the first value of the ordered pair is P_{jy} is $(I_{jy}, \text{Path\_set}_{jy}, I\_\text{Path\_set}_{jy})$, where $1 \le y \le b$. Let

$$PS_1 = \text{Path\_set}_{i1} \diamondsuit \text{Path\_set}_{i2} \diamondsuit \cdots \diamondsuit \text{Path\_set}_{ia} = \{\text{path}_{11}, \text{path}_{12}, \cdots, \text{path}_{1r}\},$$

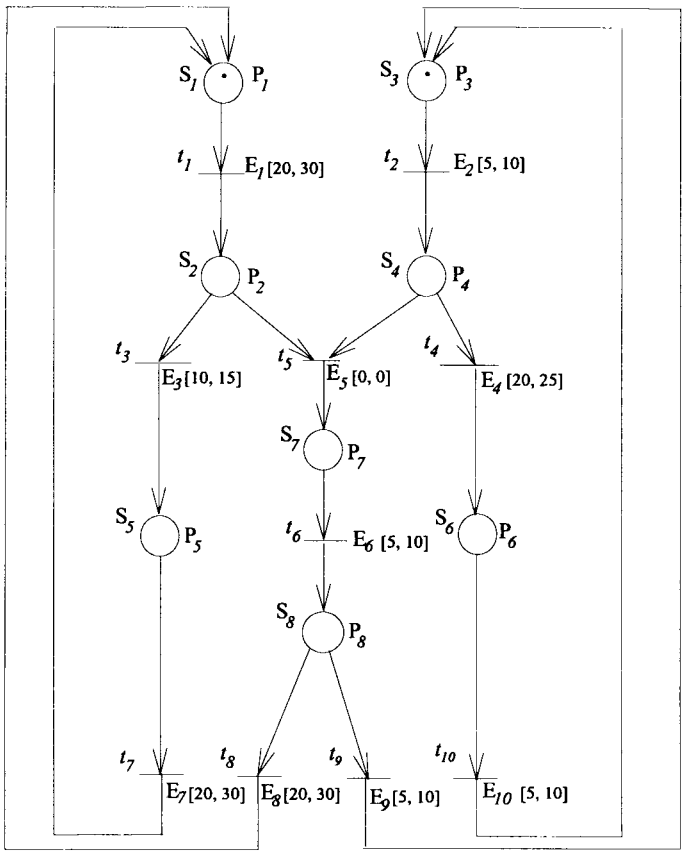

Fig. 2. Marked time Petri net of Example 4.1.

$$
\begin{aligned}
TS_1 &= I\_\text{Path\_set}_{i1} \otimes I\_\text{Path\_set}_{i2} \otimes \cdots \otimes I\_\text{Path\_set}_{ia} \\
&= \{T_{11}, T_{12}, \cdots, T_{1r}\}, \text{ where } T_{1g} = [x_{g1}, x_{g2}], \\
&\qquad \text{for } g = 1, 2, \cdots, r, \\
PS_2 &= \text{Path\_set}_{j1} \diamondsuit \text{Path\_set}_{j2} \diamondsuit \cdots \diamondsuit \text{Path\_set}_{jb} \\
&= \{\text{path}_{21}, \text{path}_{22}, \cdots, \text{path}_{2s}\}, \\
TS_2 &= I\_\text{Path\_set}_{j1} \otimes I\_\text{Path\_set}_{j2} \otimes \cdots \otimes I\_\text{Path\_set}_{ja} \\
&= \{T_{21}, T_{22}, \cdots, T_{2s}\}, \text{ where } T_{2h} = [y_{h1}, y_{h2}], \\
&\qquad \text{for } h = 1, 2, \cdots, s.
\end{aligned}
$$

If there exists a time interval T_{1g} of TS_1 and a time interval T_{2h} of TS_2, where $T_{1g} = [x_{g1}, x_{g2}]$ and $T_{2h} = [y_{h1}, y_{h2}], 1 \le g \le r$, and $1 \le h \le s$, such that $[x_{g1}, x_{g2} + i_2] \cap [y_{h1}, y_{h2} + j_2] \neq \phi$, then t_i and t_j can occur at the same time. Otherwise, t_i and t_j can't occur at the same time. If the result is that t_i and t_j can occur at the same time, then the events (or transitions) that make t_i and t_j occur at the same time are $\text{path}_{1g} \boxdot \text{path}_{2h}$, where $1 \le g \le r$, and $1 \le h \le s$.

Example 4.1: John spends 20–30 min going to school from his home by bus. Mary spends 5–10 min walking to school from her home. John reads the newspaper for 10–15 min in his classroom or talks with Mary for 5–10 min in the corridor of the school. Mary studies for 20–25 min in her classroom or talks with John for 5–10 min in the corridor of the school. After reading the newspaper or talking, John spends 20–30 min by bus from the school to his home. After studying or talking, Mary spends 5–10 min walking home from the school. Then, the following temporal knowledge can be obtained:

"state S_1 (John in his home) by event E_1 (by bus) to state S_2 (John in his classroom),"

"state S_3 (Mary in her home) by event E_2 (walking) to state S_4 (Mary in her classroom),"

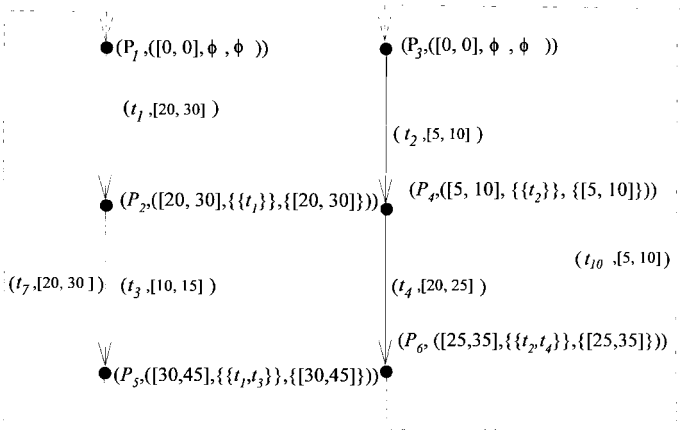

Fig. 3. Sprouting graph of Example 4.1.

"state S_2 by event E_3 (reading the newspaper) to state S_5 (John in his classroom),"

"state S_4 by event E_4 (studying) to state S_6 (Mary in her classroom),"

"(state S_2 and state S_4) by event E_5 (nothing) to state S_7 (John and Mary in the corridor),"

"state S_7 by event E6(talking) to state S_8 (John and Mary in the corridor),"

"state S_5 by event E_7 (by bus) to state S_1,"

"state S_8 by event E_8 (by bus) to state S_1,"

"state S_6 by event E_9 (walking) to state S_3,"

"state S_8 by event E_{10} (walking) to state S_3,"

where the time intervals associated with the events $E_1, E_2, E_3, E_4, E_5, E_6, E_7, E_8, E_9$, and E_{10} are [20, 30], [5, 10], [10, 15], [20, 25], [0, 0], [5, 10], [20, 30], [20, 30], [5, 10], and [5, 10], respectively, and the initial marking places are P_1 and P_3.

Based on [10], we can construct the corresponding time Petri net as shown in Fig. 2. Then, the sprouting graph can be obtained by applying the algorithm described above, where the sequence generated in Step 1 is $t_1t_2t_3t_5t_4t_6t_7t_8t_9t_{10}$, and the sprouting graph of Example 4.1 is shown in Fig. 3. From Fig. 3, we can see that the time Petri net shown in Fig. 2 is not consistent due to the fact that t_6 will not be enabled to fire. In other words, John has no chance to talk with Mary in the corridor even if indeed there is the time fact that John and Mary talk with each other. Furthermore, we know that t_7 and t_{10} can occur at some time because $[30, 45 + 30] \cap [25, 35 + 10] \neq \phi$. To make t_7 and t_{10} occur at the same time, we must take the path $\{t_1, t_3\} \boxdot \{t_2, t_4\} = \{t_1, t_3, t_2, t_4\}$. In other words, if John wants to go home with Mary at the same time, John needs to go to school by bus and read the newspaper in his classroom and Mary needs to walk to school and study in her classroom.

V. Conclusions

In this paper, we have presented temporal knowledge representation and reasoning techniques using time Petri nets. Furthermore, we also presented a method to check the consistency of the temporal knowledge. The proposed method can overcome the drawback of the one presented in [16] due to the fact that the proposed method can check the consistency of the temporal knowledge correctly.

References

[1] F. Allen, "Maintaining knowledge about temporal intervals," *Commun. ACM*, vol. 26, pp. 832–843, Nov. 1983.

[2] S. I. Ahson, "Petri net models of fuzzy neural networks," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 926–932, June 1995.

[3] S. M. Chen, "A new approach to handling fuzzy decision making

problems," *IEEE Trans. Syst., Man, Cybern.*, vol. 18, pp. 1012–1016, Nov./Dec. 1988.
[4] S. M. Chen and W. T. Jong, "Comments on "A Petri net model for temporal knowledge representation and reasoning," " *IEEE Trans. Syst., Man, Cybern B*, vol. 27, pp. 165–166, Feb. 1997.
[5] S. M. Chen, J. S. Ke, and J. F. Chang, "Knowledge representation using fuzzy Petri nets," *IEEE Trans. Knowl. Data Eng.*, vol. 2, pp. 311–319, Sept. 1990.
[6] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint network," *Artif. Intell.*, vol. 49, no. 1, pp. 61–95, 1991.
[7] Y. Deng and S. K. Chang, "A G-net model for knowledge representation and reasoning," *IEEE Trans. Knowl. Data Eng.*, vol. 2, pp. 295–310, Sept. 1990.
[8] S. Dutta, "An event based fuzzy temporal logic," in *Proc. 18th Int. Symp. Multiple-Valued Logic*, Palma De Mallorca, Spain, May 1988, pp. 64–71.
[9] M. L. Garg, S. I. Ahson, and P. V. Gupta, "A fuzzy Petri net for knowledge representation and reasoning," *Inf. Process. Lett.*, vol. 39, pp. 165–171, 1991.
[10] W. T. Jong, Y. S. Shiau, Y. J. Horng, H. H. Chen, and S. M. Chen, "Temporal knowledge representation using time Petri nets," in *Proc. 7th Int. Conf. Information Management*, Chungli, Taiwan, R.O.C., May 1996., vol. 1, pp. 312–321.
[11] G. Juanole and J. L. Roux, "On the pertinence of the extended time Petri net model for analyzing communication activities," in *Proc. 3rd Int. Workshop Petri Nets Performance Model*, Kyoto, Japan, 1989, pp. 230–235.
[12] H. Kautz and P. Ladkin, "Integrating metric and qualitative temporal reasoning," in *Proc. AAAI-91*, Anaheim, CA, 1991, pp. 241–246.
[13] D. L. Mon, C. H. Cheng, and H. C. Lu, "Application of fuzzy distributions on project management," *Fuzzy Sets Syst.*, vol. 73, pp. 227–234, July 1995.
[14] W. Pedrycz and F. Gomide, "A generalized fuzzy Petri net model," *IEEE Trans. Fuzzy Syst.*, vol. 2, pp. 295–301, Nov. 1994.
[15] J. L. Peterson, *Petri Nets, Theory, and the Modeling of Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1981.
[16] Y. Yao, "A Petri net model for temporal knowledge representation and reasoning," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, pp. 1374–1382, Sept. 1994.
[17] L. A. Zadeh, "Fuzzy logic," *IEEE Computer*, vol. 21, pp. 83–91, Apr. 1988.

Comments on "A New Approach to Adaptive Fuzzy Control: The Controller Output Error Method"

Donald S. Reay

***Abstract*—In the above paper, a novel algorithm for adaptively updating the parameters of a fuzzy controller was proposed. The purpose of this letter is to point out that this algorithm, and its use, are well known. The authors of the above paper acknowledge the previous use of similar concepts, however this letter draws attention to a particularly clear description of the algorithm.**

***Index Terms*—Adaptive control, fuzzy systems.**

I. Introduction

An algorithm for the fine tuning of the parameters of a fuzzy controller, on-line, and without the need for an inverse model of the controlled plant is proposed in the above paper [1]. The algorithm is described as novel but, in fact, both the algorithm and its use are reported widely in the literature on learning control. This letter draws attention to a particularly clear description of the algorithm, several reported examples of its use, and its characterization within a taxonomy of learning control systems.

Manuscript received October 16, 1998. This paper was recommended by Associate Editor A. Kandel.

The author is with the Department of Computing and Electrical Engineering, Heriot-Watt University, Edinburgh EH14 4AS, U.K.

Publisher Item Identifier S 1083-4419(99)05279-6.

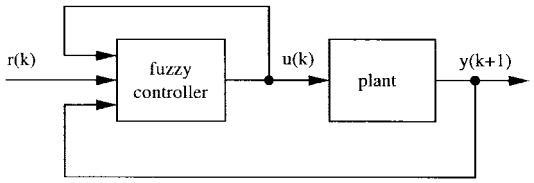

Fig. 1. Fuzzy control system suitable for the use of COEM.

II. Algorithm Description

The controller output error method (COEM) is a method of fine tuning the parameters of a fuzzy system within the control architecture shown in Fig. 1. Note that the block labeled fuzzy controller in Fig. 1 represents the combination of delay lines, fuzzy system, and learning algorithm described in the aforementioned paper.

The algorithm is described in the above paper as follows: "At instant k, the state of the plant may be defined by $S = [y(k), \cdots, y(k-p+1)]^T$ (assuming that the plant is observable). The fuzzy controller produces a control signal, $u(k)$, which drives the output of the plant to $y(k+1)$. Regardless of whether or not this was the intended response, we now know that, if the transition from a state S to an output $y(k+1)$ is ever required again, the appropriate control signal is $u(k)$.

The fuzzy controller is now tested to see if it does indeed output a signal equal to $u(k)$ when required to drive the plant through this same transition. Instead of producing a control signal $u(k)$, however, the controller outputs the signal $\hat{u}(k)$. Thus, the controller output is in error by $e_u(k) = u(k) - \hat{u}(k)$.

It is important to note that, although $\hat{u}(k)$ is produced by the controller, it is not applied to the plant. Its only purpose is to calculate $e_u(k)$. $\hat{u}(k)$ is calculated by producing a new controller input vector, $\hat{z}(k)\cdots$. The input vector $\hat{z}(k)$ only differs from $z(k)$ in the first element, where $y(k+1)$ replaces $r(k)$." The last sentence of the description refers to two alternative input vectors to a fuzzy system, $z(k) = [r(k), y(k), \cdots, y(k-n+1), u(k-1), \cdots, u(k-m)]^T$ and $\hat{z}(k) = [y(k+1), y(k), \cdots, y(k-n+1), u(k-1), \cdots, u(k-m)]^T$.

While the authors of the foregoing acknowledge the previous use of similar concepts, for example in [2], apparently they are unaware of the following description of the same algorithm by Albus [3].

"Ordinarily the CMAC training algorithm proceeds by 1) observing an input $S = (s_1, s_2, s_3, \cdots, \dot{x}, \dot{y}, \dot{z})$; 2) computing an output $P = h(S)$; 3) comparing P against a desired $\hat{P}$; and 4) adjusting weights so as to null the difference. In the process of training, the function h is modified to h' such that $\hat{P} = h'(S)$. The critical factor in this conventional technique is finding the desired output $\hat{P}$ corresponding to the actual input S. In the time inversion technique this process is inverted, i.e., the computed output P is assumed to be the *desired* output for some unknown input $\hat{S}$. The problem then is not to find the desired output $\hat{P}$ corresponding to some actual input S, but instead to find some input $\hat{S}$ for which P is the desired output. This may be done in the following manner.

First, apply the computed output P to the joint actuators and observe the resulting movement $\hat{\dot{x}}, \hat{\dot{y}}, \hat{\dot{z}}$. Now, if the original input S had called for the observed movement $\hat{\dot{x}}, \hat{\dot{y}}, \hat{\dot{z}}$ instead of $\dot{x}, \dot{y}, \dot{z}$, then P would have been exactly the correct output. Therefore, the input $\hat{S}$ for which P is the desired output, is merely the original