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 (FNM) 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.

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 $(S, E, P, T, B, F, M_0, \alpha, \beta, \text{SIM})$, where

- $S$ is a finite set of states, $S = \{S_1, S_2, \ldots, S_n\}$; 
- $E$ is a finite set of events, $E = \{E_1, E_2, \ldots, E_m\}$, where each event is associated with a transition; 
- $P$ is a finite set of places, $P = \{P_1, P_2, \ldots, P_n\}$, where each place is associated with a state; 
- $T$ is a finite set of transitions, $T = \{t_1, t_2, \ldots, t_m\}$, where each transition is associated with a time interval; 
- $B$ is a backward incidence function, $B: T \times P \rightarrow N$, where $N$ is the set of nonnegative integers; 
- $F$ is a forward incidence function, $F: T \times P \rightarrow N$; 
- $M_0$ is an initial marking function, $M_0: P \rightarrow \mathbb{N}$; 
- $\alpha$ is a mapping function from places to states, $\alpha: P \rightarrow S$; 
- $\beta$ is a mapping function from events to transitions, $\beta: E \rightarrow T$; 
- SIM is a 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 $\lambda$, 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 $[t_1, t_2], [t_3, t_4], [t_5, t_6],$ and $[t_7, t_8]$, 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 \( P_0 \) be a place, and let \( t_i \) and \( t_j \) be transitions in a time Petri net. If \( P_0 \) is a forward incidence place of \( t_i \) and \( P_0 \) 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 \( c < a \leq d \leq b \), then \( T_1 \cup T_2 = [c, b] \).
   - Case 4: If \( d < a \), then \( T_1 \cup T_2 = [a, d] \cup [c, b] \).

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 \( c < a \leq d \leq b \), then \( T_1 \cap T_2 = [c, a] \).
   - 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 \( \{t_1, t_2, \ldots, t_j\} \) 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 \( p_i \) and \( p_j \) be two paths in a time Petri net, where \( p_i = \{t_1, t_{p_1+1}, \ldots, t_{p_n} \} \) and \( p_j = \{t_1, t_{q_1+1}, \ldots, t_{q_n}, \ldots, t_{q_l} \} \). If \( t_a \) and \( t_b \) are not contradictory, then the union of the two paths (i.e., \( p_i \uplus p_j \)) is equal to \( \{t_1, t_{p_1+1}, \ldots, t_{p_n}, \ldots, t_{q_l}, \ldots, t_{q_l} \} \cup \{t_1, t_{p_1+1}, \ldots, t_{p_n} \} \cup \{t_1, t_{p_1+1}, \ldots, t_{p_n} \} \cup \ldots \).
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 $i$th path in the set of paths is $\phi$, then the $i$th time interval of the set of time intervals is $\phi$ 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 $F_2(P_i, t_j) = 0$.

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 $t_j$. Let $F_2(P_i, t_j) = 0$ 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 places in which the first value of the ordered pair associated with each place is one of $t_j$’s backward incidence places. Assume that the triplet of the ordered pairs of these nodes are $(I_1, \text{Path}, I_2, \text{Path}, I_3, \text{Path}, \ldots, I_r, \text{Path})$, where $I_i$ is a time interval, $\text{Path}$ is a set of paths, and $I_r, \text{Path}$ is a set of time intervals. Let $I = (I_1 \cap I_2 \cap \cdots \cap I_r) + T_a$.

\[ \text{Path} = \langle \text{Path}_{\text{set}} \rangle \oplus \text{Path}_{\text{set2}} \oplus \cdots \oplus \text{Path}_{\text{setx}}. \]

\[ I_{\text{Path}} = (I_{\text{Path}}_{\text{set}}) \oplus I_{\text{Path}}_{\text{set2}} \oplus \cdots \oplus I_{\text{Path}}_{\text{setx}} \oplus T_a. \]

If $(I_1 \cap I_2 \cap \cdots \cap I_r) = \phi$ (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 $P_1, P_2, \ldots, P_s$. For each $P_r, 1 \leq i \leq s$,

   Case 1: If $P_r$ 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 $P_r$ 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 $P_r$ and assume that the triplet of the ordered pair associated with the node is $(I_r, \text{Path}_{\text{set}})$, then direct 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 $(I \cup I_r, \text{Path}_{\text{set}})$.

   Case 3: If we can’t find any node in which the first value of the ordered pair associated with this node is $P_r$, then create a new node, and let the ordered pair associated with the created node be $(P_r, (I, \text{Path}_{\text{set}}))$, and direct 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 $(P_r, (I_r, \text{Path}_{\text{set}}, I_{r-1}, \text{Path}_{\text{set}}))$, where $I_r$ is a time interval, $\text{Path}_{\text{set}}$ is a set of paths, and $I_{r-1}, \text{Path}_{\text{set}}$ is a set of time intervals. Furthermore, assume that $I_{r-1}, \text{Path}_{\text{set}} = \{T_1, T_2, \cdots, T_k\}$, where $k$ is a positive integer and $P_r$ is not an initial marking place, then we can find that $I_r = T_1 \cup T_2 \cdots \cup T_k$.

In the following, we present a method to check the consistency of the temporal knowledge and to perform temporal reasoning. If thereexists 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 

\[ t_i, t_j, \cdots, t_k, \] \n
the backward incidence places of $t_i$ are $P_{r_{ij}}, P_{r_{ij}}, \ldots, P_{r_{ik}}$, and assume that the triplet associated with the node of which the first value of the ordered pair is $P_{r_{ij}}$ is $(I_{r_{ij}}, \text{Path}_{\text{set}}_{r_{ij}}, I_{r_{ij}} \text{Path}_{\text{set}}_{r_{ij}})$, where $1 \leq y \leq k$. The time interval associated with $t_j$ is $(t_j, t_j, \cdots, t_j)$, the backward incidence places of $t_j$ are $P_{r_{ij}}, P_{r_{ij}}, \ldots, P_{r_{jk}}$, and assume that the triplet associated with the node of which the first value of the ordered pair is $P_{r_{ij}}$ is $(I_{r_{ij}}, \text{Path}_{\text{set}}_{r_{ij}}, I_{r_{ij}} \text{Path}_{\text{set}}_{r_{ij}})$, where $1 \leq y \leq k$. Let

\[ P_{s_{ij}} = \text{Path}_{\text{set}}_{r_{ij}} \oplus \text{Path}_{\text{set}}_{r_{ij}} \oplus \cdots \oplus \text{Path}_{\text{set}}_{r_{ij}} \]

\[ = \{p_{1_{ij}}, p_{2_{ij}}, \cdots, p_{k_{ij}}\}. \]
If there exists a time interval $[5, 10]$, respectively, and the initial marking places are $P_1$ and $P_5$. 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 $t_1t_2t_3t_4t_5t_6t_7t_8t_9t_{10}$, 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 $t_9$ 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 $t_2$ and $t_{10}$ can occur at some time because $[30, 45] \cap [25, 35] \neq \emptyset$. To make $t_2$ and $t_{10}$ occur at the same time, we must take the path $\{t_2, t_5\}$ and $\{t_{10}, t_3\}$. 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**


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, 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 \( S = (s_1, s_2, \ldots, \hat{x}, \hat{y}, \hat{z}) \); 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 inverse time inversion 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 \( \hat{x}, \hat{y}, \hat{z} \). Now, if the original input \( S \) had called for the observed movement \( \hat{x}, \hat{y}, \hat{z} \) instead of \( \hat{x}, \hat{y}, \hat{z} \), 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

<table>
<thead>
<tr>
<th>Reference</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>The algorithm described in the above paper is 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.</td>
</tr>
<tr>
<td>[2]</td>
<td>The algorithm described in the above paper as follows: “At instant ( k ), the state of the plant may be defined by ( S = [y(k), \ldots, 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) ).”</td>
</tr>
<tr>
<td>[3]</td>
<td>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) = u(k) - \hat{u}(k) ). 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, \ldots, \hat{x}, \hat{y}, \hat{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 inverse time inversion 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.”</td>
</tr>
</tbody>
</table>