Proceedings of the 27th Conference
on Decision and Control
Austin, Texas • December 1988

FORMULATION AND CONTROL OF REAL TIME DISCRETE EVENT PROCESSES

Y. Brave
Department of Electrical Engineering
Technion - Israel Institute of Technology
Haifa 32000, Israel.
M. Heymann[1]
Department of Computer Science
Technion - Israel Institute of Technology
Haifa 32000, Israel.

Abstract

A discrete event process is modelled as a controlled state machine, in the framework of Ramadge and Wonham. Their approach is extended to model a class of real-time discrete event processes by means of a special type of automaton, called a clock automaton. The clock automaton is used for describing the real time behavior of processes, controllers and real time specifications.

1. Introduction

Discrete-event processes (DEP) are processes in which state transitions take place discretely, asynchronously (i.e. without reference to a clock) and sometimes nondeterministically. Such processes are used to describe computer communication networks, flexible manufacturing systems, traffic networks and the like. Ramadge and Wonham (RW) in a series of papers [1-3] introduced a framework for modeling and controlling the behavior of DEPs. In their framework the DEP is modeled as an automaton that executes asynchronous state transitions some of which can be disabled by a suitable feedback control mechanism so as to confine the DEP's behavior to within specified (legal) bounds.

In the present paper the RW approach is extended to model *real time discrete event processes* (RTDEP), that is, to DEP's whose state transitions depend not only on the current state and transition symbol, but also on the current value of a clock or counter (that represents, for example, the time that elapsed since the current state was entered). A more complete and precise discussion of the model is currently in preparation [4].

2. Real-Time Processes

A RTDEP is modeled as a special type of automaton that we call a *clock automaton*

$$P=(Q,\Sigma,\delta,q_o,Q_m,C,u)\ .$$

Here, just as in RW, Q is a state set, Σ is a transition alphabet, q_o is the initial state and Q_m a set of final or marker states. The new element is C, a *counter* whose current value is $c\in \mathbf{Z}^+$, the set of nonnegative integers. The counter is endowed with a counter update map which at each 'tick' of the real-time clock updates the counter value as follows: If a transition ($\sigma\in\Sigma$) is recorded (i.e., a state transition occurs) the counter is reset to zero. Otherwise (denoted by the symbol τ and termed the *null-event*) the counter is incremented by a unit. It is assumed that the 'ticking' of the clock is frequent enough so as to ensure that at most one state transition can occur within a single time unit. Formally, the counter operation is thus described by the *update map* $u:(\Sigma\cup\{\tau\})\times\mathbf{Z}^+\to\mathbf{Z}^+$, where

$$u(\sigma,c)=\begin{cases} c+1 & \text{if } \sigma=\tau \\ 0 & \text{otherwise}\end{cases}\ .$$

For abbreviation let $\Sigma_\tau:=\Sigma\cup\{\tau\}$. We define the *configuration* of P as the pair $(c,q)\in\mathbf{Z}^+\times Q$. The symbol δ represents the *state transition function* $\delta:\Sigma_\tau\times\mathbf{Z}^+\times Q\to Q$. The transition function satisfies the condition that $\delta(\tau,c,q)=q$ for all $(c,q)\in\mathbf{Z}^+\times Q$. In general, however, δ need not be defined for all $(\sigma,c,q)\in\Sigma\times\mathbf{Z}^+\times Q$ and, thus, is a partial function. It is also convenient to define the *configuration transition function* (partial function) $g:\Sigma_\tau\times\mathbf{Z}^+\times Q\to\mathbf{Z}^+\times Q$ as $g(\sigma,c,q):=(u(\sigma,c),\ \delta(\sigma,c,q))$ whenever $\delta(\sigma,c,q)$ is defined.

We interpret the RTDEP as a device that starts at configuration $(0,q_o)$ and executes successive configuration transitions according to its configuration transition function. Configuration transitions occur instantaneously at (normalized) discrete times, and with each transition is associated a symbol $\sigma\in\Sigma_\tau$.

Let Σ_τ^* denote the set of all finite strings s of elements of Σ_τ, including the empty string ε. Then $g:\Sigma_\tau\times\mathbf{Z}^+\times Q\to\mathbf{Z}^+\times Q$ is extended in a natural way to $g:\Sigma_\tau^*\times\mathbf{Z}^+\times Q\to\mathbf{Z}^+\times Q$.

3. The Languages of P

3.1 Execution-language

Let $\bar{\Sigma}_\tau^*\subset\Sigma_\tau^*$ denote the subset consisting of the empty string ε and all strings of Σ_τ^* that end with a symbol $\sigma\neq\tau$. A subset $L\subset\bar{\Sigma}_\tau^*$ is called an *execution-language* over Σ_τ and every string $s\in L$ is called an *execution-word*. The execution-language generated by P, denoted $L_e(P)$, is defined as

$$L_e(P)=\{\,s\in\bar{\Sigma}_\tau^*:\ g(s,0,q_o)\ \text{ is defined}\,\}\ .$$

3.2 Event-language

It will be more convenient to represent execution-words in the following way. Let $s\in\bar{\Sigma}_\tau^*$ be a nonempty execution-word, and partition it into a sequence of successive nonempty substrings such that in every substring there is exactly one symbol $\sigma\neq\tau$ which is the last element of the substring. Each such substring is called an *event*, which alternatively can be represented by $e:=(\sigma,c)\in\Sigma\times\mathbf{Z}^+$, where c is the number of τ-symbols in the substring. With this notation, an execution-word is a finite string of events and it is readily verified that in each event (σ,c) the number c is exactly the counter value just before the associated configuration transition labeled σ occurs.

Let $E=\Sigma\times\mathbf{Z}^+$ be the set of all events, and let E^* denote the set of all finite strings of elements of E including the empty string ε. A subset $L\subseteq E^*$ is called an *event-language* over Σ and every string $s\in L$ is called an *event-word*. The closure of L, denoted $\bar{L}$, is the set of all strings (in E^*) that are prefixes of event-words of L. An event-language L is closed if $L=\bar{L}$.

In standard fashion we can extend the transition function $\delta:E\times Q\to Q$ to $\delta:E^*\times Q\to Q$. The event-language $L(P)$ generated by P is then defined as the subset of E^* consisting of all strings s in E^* for which $\delta(s,q_o)$ is defined.

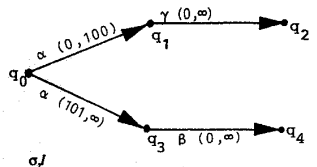
The process P can also be interpreted as a device that starts at state q_o and executes *state* transitions, i.e. generates a sequence of *events* according to its transition function. With our second interpretation we can view P as a directed graph with node set Q. An edge $q\to q'$ exists and is labeled $e\in E$ provided the triple (e,q,q') satisfies the condition that $q'=\delta(e,q)$.

To illustrate the expressive power of our real time discrete event process model let us consider the following simple examples.

[1] Supported in part by the Technion Fund for Promotion of Research.

Example 3.1

Let P be a RTDEP over $\Sigma = \{ \alpha, \beta, \gamma \}$ displayed below.

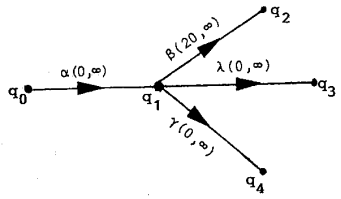


σ,I

Here the notation $q \to q'$, $\sigma \in \Sigma$, $I \subset \mathbf{Z}^+$, represent the fact that the transition (σ, q, q') can occur only if $c \in I$. Thus, the event γ will follow α provided α occurs within the first 100 ticks of the clock; otherwise α will be followed by β.

Example 3.2

Let P be a RTDEP over $\Sigma = \{ \alpha, \beta, \gamma, \lambda \}$ displayed below.



In this example, the events γ and λ can occur at any time after entering state q_1 while β becomes possible only after 20 ticks of the clock. Thus, the notation $\sigma(t_1,t_2)$ can also model a minimal delay between successive events. Suppose that β is an undesirable event and that the event γ can be forced by an external agent. Then in order to prevent β from occurring a controller for P must enforce γ within 20 time units unless λ occurs within that time interval. In the following we discuss the existence of appropriate controllers (and their structure) that will satisfy such and other requirements. In addition, our real-time model can be used for describing real-time specifications.

4. Control Mechanism

As in [5] we shall view the transition alphabet Σ of a process P as consisting of three subsets Σ_u, Σ_c and Σ_f that are called, respectively, the *uncontrolled*, *controlled* and *forced alphabet* of P. A spontaneous transition labeled $[\sigma, I], \sigma \in \Sigma_u \cup \Sigma_c, I \subset \mathbf{Z}^+$ *can* be generated by P only if $c \in I$, and can be disabled only if $\sigma \in \Sigma_c$. A transition labeled $[\sigma, I]$, $\sigma \in \Sigma_f, I \subset \mathbf{Z}^+$, will occur only if (and when) it is forced provided $c \in I$. It is assumed that such transitions *occur* instantaneously (whenever they are forced) and prevent any other event from occurring at the same time.

Formally let $\Gamma \subseteq 2^{\Sigma}$ be defined as

$$\Gamma = \{ \gamma \in 2^{\Sigma} : \text{either } \gamma = \{ \sigma \} \text{ for some } \sigma \in \Sigma_f \quad \text{or} \quad \Sigma_u \subseteq \gamma \subseteq \Sigma_u \cup \Sigma_c \} .$$

We interpret a element $\gamma \in \Gamma$ as the 'next' state transitions that may (or must) occur within a time unit. Now we define a *real-time control pattern* to be a map $\lambda : \mathbf{Z}^+ \to \Gamma$ that satisfies the condition that: for given $c \in \mathbf{Z}^+$, $\lambda(c)$ is defined if and only if for no $c' \in \mathbf{Z}^+$ such that $c' < c$

$$\lambda(c') = \{ \sigma \} \quad , \quad \sigma \in \Sigma_f .$$

Let Λ denote the set of all real-time control patterns. With each $\lambda \in \Lambda$ is associated a subset $\lambda_a \subseteq E (= \Sigma \times \mathbf{Z}^+)$ defined as

$$\lambda_a = \{ e := (\sigma, c) : e \in E \quad \text{and} \quad \sigma \in \lambda(c) \} .$$

The associated control pattern can be interpreted as the set of all possible 'next' events in the evolution of the RTDEP.

5. Real-Time Supervisors

A *real-time supervisor* (RTS) $S = (\underline{S}, \phi)$ consists of a clock automaton $\underline{S} = (X, \Sigma, \xi, x_o, X_m, C, u)$ together with a feedback map $\phi : X \to \Lambda$. The process P is constrained (and triggered) by the real-time control determined by the states of $\underline{S}$. The transitions of the closed loop system S/P are given by

$$(\sigma, c, x, q) \mapsto \begin{cases} (\xi(\sigma, c, x), \delta(\sigma, c, q)) & \text{if } \sigma \in [\phi(x)](c) \\ \text{undefined} & \text{otherwise} \end{cases}$$

whenever $\xi(\sigma, c, x)$, $\delta(\sigma, c, q)$ and $[\phi(x)](c)$ are defined. Actually we will be interested only in RTS whose transitions are defined whenever they can occur in P and allowed by ϕ. Further, we shall require that whenever forced events are triggered in S they are defined in P. Such RTS are called well posed (with respect to P).

6. Real-Time Controllability

In this section we characterize the event-language that can be generated by the closed loop S/P that consists of a given RTDEP P and a well posed RTS S. Let $K \subseteq E^*$ be an arbitrary event-language. For each $s \in \overline{K}$ we define

$$E_K(s) = \{ e \in E = \Sigma \times \mathbf{Z}^+ : se \in \overline{K} \} .$$

That is, $E_K(s)$ is the set of all events of the form $e := (\sigma, c)$ that are possible after s in $\overline{K}$.

Definition 6.1: An event language $K \subseteq L(P)$ is called *real time controllable* (RTC) if for all $s \in \overline{K}$ there exists $\lambda \in \Lambda$ such that

$$\lambda_a \cap E_{L(P)}(s) = E_K(s) \quad \text{and} \quad \lambda_a \cap (\Sigma_f \times \mathbf{Z}^+) \subseteq E_{L(P)}(s)$$

Thus, if we interpret $L(P)$ as the physically possible behavior of a RTDEP P, then an event-language K is RTC if every prefix $s \in \overline{K}$ is possible and for $s \in \overline{K}$ either

(i) $E_K(s)$ does not contain any forced event and every physically possible string se, with e uncontrolled, is again in $\overline{K}$; or

(ii) $E_K(s)$ contains exactly one forced event $(\sigma_f, c_f) \in \Sigma_f \times \mathbf{Z}^+$, every physically possible string se with $e := (\sigma, c)$, $\sigma \in \Sigma_u$, $c < c_f$ is again in $\overline{K}$, and

$$(\sigma, c) \notin E_K(s) , \qquad \sigma \neq \sigma_f, c = c_f \quad \text{or} \quad \sigma \in \Sigma, c > c_f$$

Proposition 6.1

Let $K \subseteq L(P)$ be an event-language. There exists a well posed RTS S such that $L(S/P) = K$ if and only if K is closed and RTC.

Other issues as controllability and timing information, minimally restrictive RTS and alternative counter update maps are discussed in [4]. Related work on real time issues in control discrete event processes is the work of Ostroff and Wonham [6] that deals with real time temporal logic and the work of Yong and Wonham [7] that deals with communication delays and simultaneity of events.

REFERENCES

[1] Ramadge, P. J. and W. M. Wonham, "Supervisory control of a class of discrete event processes", SIAM J. on Control and Optimization, 25(1), pp. 206-230,January 1987.

[2] Ramadge, P.J. and W. M. Wonham, "Modular supervisory control of discrete event systems", Proc. 7th Int. Conf. on Analysis and Optimization of Systems, Antibes, June 1986.

[3] Wonham, W. M. and P. J. Ramadge, "On the supremal controllable sublanguage of a given language", SIAM J. Control and Optimization 25(3), pp. 637-659,May 1987.

[4] Brave, Y. and M. Heymann, " Real time discrete event processes ", preprint 1988.

[5] Golaszewski, C.H. and P.J. Ramadge, "Control of discrete event processes with forced events", Proc. 26th IEEE Conf. on Decision and Control, Los Angeles, pp. 247-251,December 1987.

[6] Ostroff, J.S. and W. M. Wonham, "State machines, temporal logic and control: a framework for discrete event systems", Proc. 26th IEEE Conf. on Decision and Control, IEEE Control Systems Society, Los Angeles, pp. 656-657,December 1987.

[7] Yong, L. and W. M. Wonham, "On supervisory control of real-time discrete event systems", Proc. American Control Conference, Minneapolis, pp. 1715-1720,June 1987.