Proceedings of the 30th Conference
on Decision and Control
Brighton, England • December 1991

The Control of Dense Real-Time Discrete Event Systems

H. Wong-Toi [1], G. Hoffmann [2]
Stanford University, Stanford, CA 94305

Abstract— **We introduce *dense real-time* into the supervisory control of discrete event systems. We give conditions for the existence of a controller. If the plant and specification behaviors are represented by timed automata [1, 2], there is a supremal controllable sublanguage of the specification language for a subclass of synthesis problems. The synthesized supervisor is polynomial in the number of automata states and exponential in the timing information.**

I. Introduction

We extend the framework of Ramadge and Wonham [9, 8] to *timed* execution sequences. The correct behavior of *hard real-time systems* (*e.g.* manufacturing systems) depends on the actual delays between events. Previously, Brave and Heymann [3] and Golaszewski and Ramadge [4] used the domain of integers to model time. These *discrete-time* models specify *a priori* the smallest measurable time unit. The *fictitious clock* approach makes time a global state by including an explicit tick transition [7]. Events between the i-th and $i+1$-th clock ticks are assumed to occur at some unspecified time between time i and $i+1$. Thus the exact time delay between any two events is unknown.

In this paper we use timed traces defined over a *dense* domain of time. An exact occurrence time is associated with each event [1, 2]. We use Alur and Dill's *timed automata* to describe the behavior of timed discrete event systems. Any number of system timers can be accommodated. The automata used by Brave and Heymann, and Golaszewski and Ramadge are a special case of timed automata with a single system timer interpreted over the integers.

We consider the control of both *finite* and *infinite* executions. Infinite executions model nonterminating processes. We can then reason about the limiting behavior of a system, such as the fair composition of concurrent components. Various researchers have considered supervisory control over infinite strings [8, 10, 5].

Due to space limitations, we present only the main results without proof. A detailed presentation is under preparation.

II. Timed Traces

Our model of a discrete-event process is a set of timed traces with R_+ as our domain of time. Let Σ be a finite alphabet of events. For any set A, we abbreviate $A^* \cup A^\omega$ by A^∞, and use $len(s)$ to denote the length of any element in A^*. Elements of A^ω have length ω. For an element $s \in A^\infty$, we let s_i denote its component at the $i+1$-th position, if $0 \le i < len(s)$. The symbol λ denotes the empty string. Formally, a *timed trace* $\nu = \langle\sigma_0, t_0\rangle, \langle\sigma_1, t_1\rangle, \ldots$ is a sequence in $(\Sigma \times R)^\infty$, satisfying: (1) for $0 \le i < len(\nu), \sigma_i \in \Sigma$; (2) $t_0 = 0$; (3) for all $0 \le i < len(\nu)-1$, $t_i < t_{i+1}$, and (4) either $len(\nu)$ is finite or for all $r \in R$ there is a pair $\langle\sigma_i, t_i\rangle$ such that $t_i \ge r$. This last condition asserts that time progresses. A finite string $\mu = \langle\sigma'_0, t'_0\rangle, \ldots \langle\sigma'_n, t'_n\rangle$ in $(\Sigma \times R)^*$ is a *prefix* of the timed trace ν above if $\langle\sigma_i, t_i\rangle = \langle\sigma'_i, t'_i\rangle$ for $0 \le i < len(\mu)$. A *timed language* L is any set of timed traces. Its set of prefixes is denoted $pr(L)$. Let K and L be languages of finite traces. L is *prefix-closed* iff $L = pr(L)$. K is *L-closed* iff $pr(K) \cap L = K$. The *closure* of L, denoted L^∞, is the set of all infinite timed traces with infinitely many prefixes in L. For languages B and P of infinite traces, B is *closed relative to P* if and only if $pr(B)^\infty \cap P \subseteq B$.

III. Real-Time Controllability

The uncontrolled plant is modeled as a language of timed traces [1]. As usual, the event set Σ is partitioned into *controllable* events Σ_c and *uncontrollable* events Σ_u. Controllable events can be prevented from occurring at any time. A *control mask* γ is any subset of Σ that includes Σ_u. Applying the mask γ at time t means that every event in γ is enabled at time t. Let Γ denote the set of all control masks. Given a plant P, a supervisor f for P is a function $f : pr(P) \to [R \to \Gamma]$. We consider both the case of finite and infinite traces for plant and specification language.

The language L_0 of prefixes generated by P under f's supervision is given by: (1) $\lambda \in L_0$; (2) $w.\langle\sigma, t\rangle \in L_0$ iff $w \in L_0$, $\sigma \in f(w)(t)$ and $w.\langle\sigma, t\rangle \in pr(P)$. If the plant is modeled as a language L of finite timed traces, its supervised language L_f is $L_0 \cap L$. If $pr(L_f) = L_0$, then f is a *nonblocking* supervisor. For a plant P modeled as infinite timed traces, its supervised language P_f is defined as $pr(L_0)^\infty \cap P$. The supervisor f is *nonblocking* for P iff $pr(P_f) = L_0$.

Definition 1 *Let K and L be languages of finite timed traces such that $K \subseteq L$. K is controllable wrt. [1] L iff* $\mathrm{pr}(K).(\Sigma_u \times R) \cap L \subseteq \mathrm{pr}(K)$.

Theorem 1 *Let K and L be languages of finite timed traces such that $K \subseteq L$. There is a nonblocking supervisor f such that $L_f = K$ iff K is controllable wrt. L and K is L-closed.*

Theorem 2 *If $K \subseteq L$ then there is a unique supremal controllable sublanguage of K.*

Similarly, let B and P be languages of infinite timed traces. The following theorems are the timed counterparts of Propositions 3.1 and 3.2 given by Ramadge [8].

Theorem 3 *If $B \subseteq P$ is nonempty, then there is a nonblocking supervisor f for P such that $P_f = B$ iff $\mathrm{pr}(B)$ is controllable wrt. $pr(P)$ and B is closed relative to P.*

Theorem 4 *If $B \subseteq P$ is closed relative to P, then there is a unique supremal controllable sublanguage of B.*

IV. Timed Automata

Timed automata are standard finite-state automata augmented with real-time constraints on the delays between events [1, 2]. Each timed automaton has a set of clocks which may only be reset when transitions are made. All clocks have value 0 at the start of a run. The value of each clock records the time

[1] Department of Computer Science, Stanford University, CA 94305, e-mail: howard@cs.Stanford.EDU. Supported by NSF under grant MIP-8858807.

[2] Information Systems Laboratory, Stanford University, CA 94305, e-mail: hoffmann@isl.Stanford.EDU. Supported by DARPA under contract F49620-90-C-0014 monitored by AFOSR.

[1] The plant is assumed to be *nonblocking*.

elapsed since its last reset. A transition can only occur when the current values of the clocks satisfy its timing constraint.

A timed automaton TA is a tuple $A = \langle \Sigma, S, s_0, F, C, \delta \rangle$. It has a finite alphabet Σ and a finite set of states S, of which s_0 is the initial state. The set $F \subseteq S$ defines the final accepting states and C is a set of clocks, named $\mathbf{1}, \mathbf{2}, \ldots, \mathbf{N}$. The TA A has a set of transitions $\delta \subseteq S \times S \times (\Sigma \cup \{\epsilon\}) \times 2^N \times En$, where En is the set of enabling conditions, namely the boolean closure of the atomic conditions $x \sim c$ where x is a clock, c is an integral constant, and $\sim$ is one of $\{\leq, <, =, >, \geq\}$. If $\langle s, q, \sigma, \pi, E \rangle$ is in δ and the clocks satisfy E, then A may move from state s to state q on input σ at the same time as resetting to zero the clocks in π.

A *time assignment* is a function $v : C \rightarrow \mathrm{R}$. Let V be the set of time assignments. $[X \mapsto t]v$ is the time assignment that assigns time t to every clock in X but is otherwise the same as v.

A run of A over the timed trace $\nu = \langle \sigma_i, t_i \rangle_i \in (\Sigma \times \mathrm{R})^\infty$ is a sequence of triples, $\rho = \langle s_j, v_j, u_j \rangle_j \in (S \times V \times \mathrm{R})^\infty$ together with a sequence $\tilde{\sigma} \in (\Sigma \cup \{\epsilon\})^\infty$ such that: (1) the input is ν, *i.e.* deleting all ϵ's from $\tilde{\sigma}$ leaves the subsequence $\sigma = \tilde{\sigma}_{n_0} \tilde{\sigma}_{n_1}, \ldots$, with $\sigma_i = \tilde{\sigma}_{n_i}$ and $u_{n_i} = t_i$ for $0 \leq i < len(\sigma)$; (2) $\rho_0 = \langle s_0, 0, 0 \rangle$; (3) for all j, $u_{j+1} > u_j$; (4) for all j, there is a tuple $\langle s_j, s_{j+1}, \tilde{\sigma}_j, \pi_j, E_j \rangle$ in δ such that: (a) $v_j + u_{j+1} - u_j$ satisfies E_j, and (b) $v_{j+1} = [\pi_j \mapsto 0](v_j + u_{j+1} - u_j)$.

We consider two sorts of timed automata: acceptors of finite traces, and acceptors of infinite traces. A *timed regular automaton (TRA)* accepts ν when ν is finite and has a finite run of A with its last event ($\tilde{\sigma}_{len(\tilde{\sigma})-1} \neq \epsilon$) ending in a state in F, i.e. $s_{len(\rho)-1} \in F$. A *timed Büchi automaton (TBA)* accepts ν when $len(\nu)$ is infinite and has a run of A in which some state $s \in F$ is repeated infinitely often. Let $\mathcal{L}(A)$ denote the language accepted by A.

Let A be a deterministic timed regular automaton. A run of A depends not so much on the exact event times as on the time assignments of the clocks. Fortunately time assignments may be partitioned into a *finite* set of equivalence classes, $\mathcal{V}(A) = \{[v] \mid v \text{ a time assignment}\}$. $[v]$ denotes v's equivalence class. If A's runs $\rho = \langle s_j, v_j, u_j \rangle_j$ over the trace $\nu = \langle \sigma_i, t_i \rangle_i$ and $\rho' = \langle s'_j, v'_j, u'_j \rangle_j$ over $\nu' = \langle \sigma_i, t'_i \rangle_i$ have $[v'_j] = [v_j]$ for all j then $s'_j = s_j$ for all j. See [1] for details.

It is possible to construct an *untimed* deterministic automaton $A' = Untime(A)$ of finite strings over the enlarged alphabet $\Sigma \cup \{\epsilon\}$ that preserves the timing information of A. It accepts any trace that corresponds to a prefix of a timed trace in $L(A)$. Its state set is $S \times \mathcal{V}(A)$. Its final states F' are all states from which it is possible to reach a state in $F \times \mathcal{V}(A)$. The automaton A' is almost the same as Alur and Dill's M'' of Section 3.3 [1][2]. The automaton A' has $O(|C|! \cdot (|S| + |\delta|) \cdot 2^d)$ states, where d is the number of bits needed to encode the integer constants in the transition constraints [1].

V. Supervisor Synthesis

The supervisor synthesis problem consists of constructing a supervisor for the supremal controllable sublanguage of a given specification language. Recall that when an untimed closed plant and its specification are given as deterministic finite-state automata, the supervisor synthesis problem is polynomially solvable [9]. We show that for timed traces the problem is exponential.

[2]Since we are concerned only with finite runs and their prefixes, we need not follow Alur and Dill in converting their M'' into a Büchi automaton with liveness constraints.

A. Languages of Finite Timed Traces

The synthesis problem for timed traces can be solved by untiming the automata, and then applying the synthesis algorithm for untimed processes. However, since the timing information in the individual automata is independent, the automata must first be synchronized via a product construction.

Theorem 5 *Let K and L be languages of finite timed traces, where K is L-closed and $K \subseteq L$. Let A_K and A_L be timed automata accepting K and L. Finding a supervisor for the largest controllable sublanguage of K is solvable in time polynomial in the sizes of A_K and A_L and exponential in the total number of clocks and the length of their timing constants.*

This exponential factor is not due to the use of real numbers for times, since the problem is PSPACE-complete even over a discrete domain.

B. Languages of Infinite Timed Traces

In the untimed case, when the plant is closed relative to its specification, the infinite trace supervisory control problem reduces to that over finite traces, namely over the prefix sets of the plant and the specification [8]. The same result holds for infinite timed traces. Since it is possible to construct a timed regular automaton for the prefixes of a timed Büchi automaton, the theorem below is true.

Theorem 6 *Let B and P be languages of infinite timed traces represented by deterministic timed Büchi automata. If B is closed relative to P, then there is an algorithm to find a supervisor for the supremal controllable sublanguage of B wrt. P.*

The solution has the same complexity as for finite timed traces.

Acknowledgements: We would like to thank Rajeev Alur and David Dill for helpful discussions and suggestions.

References

[1] R. Alur and D. Dill, Automata for Modeling Real-Time Systems, In *Proceedings of the 17th International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science 443, Springer-Verlag 1990.

[2] R. Alur, C. Courcoubetis, and D. Dill, Model-Checking for Real-Time Systems, In *Proceedings of the 5th IEEE Symposium on Logic in Computer Science*, pages 414–425, 1990.

[3] I. Brave and M. Heymann, Formulation and control of Real-Time Discrete Event Processes, In *Proceedings of 27 Conf. Decision and Control*, 1988.

[4] C.H. Golaszewski and P.J. Ramadge, On the Control of Real-Time Discrete Event Systems, In *23rd Annual Conference on Information Sciences and Systems*, pages 98–102, Princeton, NJ, 1989.

[5] R. Kumar, V. Garg, and S.I. Marcus. On ω-controllability and ω−normality of DEDS. In *Proc. of 1991 American Control Conference*, Boston, MA, June 1991.

[6] Y. Li and W.M. Wonham. On supervisory control of real-time discrete event systems. In *Proceedings of American Control Conference*, pages 1715–1720, 1987.

[7] J.S. Ostroff and W.M. Wonham, A Framework for Real-Time Discrete Event Control, *IEEE Transactions of Automatic Control*, **35**(4):386–397, April 1990.

[8] P.J. Ramadge, Some Tractable Supervisory Control Problems for Discrete-Event Systems Modeled by Büchi Automata, *IEEE Transactions of Automatic Control*, **34**(1):10–19, January 1989.

[9] P.J. Ramadge and W.M. Wonham, The Control of Discrete Event Systems, *Proceedings of the IEEE*, **77**(1):81–98, January 1989.

[10] J.G. Thistle and W.M. Wonham, On the Synthesis of Supervisors Subject to ω-language Specifications, In *22nd Annual Conference on Information Sciences and Systems*, Princeton, NJ, March 1988.