Chapter 1 : CiteSeerX â€" Citation Query Supervisory Control of Discrete Event Systems Using Petri Nets

x Supervisory Control of Discrete Event Systems using Petri Nets b A siphon controlling supervisor is added to the net. 87 Transfonnation of a transition. 92 a A net that will have a firing constraint enforced.

This is an open access article distributed under the Creative Commons Attribution License , which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. Abstract Mutual exclusion problems widely exist in discrete event systems in which several processes will compete for the common resource for maintaining their normal running. This competition is mutually exclusive. However, a special behavior, that is, periodic mutual exclusion behavior, is important for many discrete event systems. Once a process obtains the common resource, it will consecutively obtain the common resource in the following several competitions. The other processes should wait for the release of the common resource. All processes will compete for the common resource again after the common resource is released. These competitions have obvious periodicity. In this paper, a methodology is proposed to design periodic mutual exclusion supervisors to control the periodic mutual exclusion behavior in discrete event systems. Moreover, two original structural conversion concepts, called -derivation and -convergence processes, are proposed to construct the periodic mutual exclusion supervisors. The discussion results show that many undesirable execution sequences are forbidden since the periodic mutual exclusion behavior is controlled by the proposed periodic mutual exclusion supervisors. Finally, an example is used to illustrate the proposed methodology. Introduction Discrete event systems [1] are classic dynamic systems, which widely exist in communication, manufacturing, traffic, and computer network fields, such as e-business systems, online game server systems, embedded systems, flexible manufacturing systems, and traffic management systems [2]. The concurrency behavior [3] of discrete event systems is to consider several events that simultaneously occur and potentially interact with each other. Concurrent events can be alternately executed on a single processor but can be also concurrently executed on several processors. The occurrence sequences of these concurrent events are important since system concurrency behavior can optimize system performances and promote the sufficient utilization for finite system resources [4]. The occurrence of some concurrent events may simultaneously occupy some special resources, such as a CPU, a printer, a robot, or a shared file. These special resources are finite because of the restriction of high implementation costs. Therefore, these concurrent events should share their special resources. This will form a competition among these concurrent events [5]. On the other hand, the competition will maximize the utilization of shared special resources. A most widespread and important phenomenon is that only one special resource can be used for several concurrent events. This results in a mutual exclusion behavior [6] for these concurrent events only one of them can obtain the special resource at the same time and the other events should wait for the release of the occupied special resource. Mutual exclusion behaviors as an important character in discrete event systems are usually used for resource allocation resource sharing or marking constraint in Petri nets [7], such as the user request distribution in a single input cluster of computer servers, the raw material distribution of a robot for two production lines, and the database access control for multiple processes. In discrete event systems, the special resource utilization rates are an important indicator to measure the performance of discrete event systems. It is necessary to design appropriate control strategies to control the special resource distribution for discrete event systems. The purpose is to maximize the utilization of shared special resources. The authors in [8] consider deterministic feasibility and time complexity of two fundamental tasks in the distributed computing of consensus and mutual exclusions. In [9], a mutual exclusion element is described using a reflective semiconductor optical amplifier and a simple scheme for contention resolution in arrayed waveguide grating router based optical switches in data centers. The authors in [10] propose an approach to compete for the privilege of passing the intersection that is a classical mutual exclusion problem via the communications among vehicles and infrastructure. In [11], two algorithms are presented to compete for exclusive access to a shared resource among geographically close nodes in a mobile ad hoc network. Supervisory control theory [12 , 13] is widely used to perform the controller syntheses of discrete event systems, which is proposed by Ramadge and Wonham based on automata [14 , 15]. In order to avoid the numerous state representations of automata, the subsequent researchers use Petri nets to study the control strategies. The authors in [16] present a framework for supervisor synthesis for discrete event systems. The approach is based on compositional minimization by using concepts of process equivalence. For the studies of control strategies in discrete event systems, maximally permissive problems should be considered. The authors in [17] present a computational method to design optimal control places in order to obtain a maximally permissive liveness-enforcing supervisor by using a vector covering approach. The authors in [18] distinguish between the offline and the online computation that is required for the effective implementation of the maximally permissive deadlock avoidance policy. The authors in [19] study the maximally permissive abstractions for the hierarchical and decentralized control of large-scale discrete event systems. Moreover, the structure optimization problems of supervisors are also important. The authors in [20] present two iterative deadlock prevention policies for flexible manufacturing systems. The proposed methods can reduce the overall computational time. The authors in [21] propose a deadlock prevention method to obtain maximally permissive supervisor and minimize its structure. The authors in [22] propose a deadlock prevention method for a flexible manufacturing system with a minimal supervisory structure. Furthermore, researchers also concern the design and implementation problems of Petri net based supervisors. In [23], the authors present the design, generation, and implementation of coordinating discrete event control code using Petri nets for an operating flexible manufacturing system. The authors in [24] use Petri nets to model the operated behaviors and to synthesize the command filters in a command filtering framework. Automation Petri nets can be used to perform the design and implementation of discrete event control systems by converting automation Petri nets into ladder diagrams on programmable logic controllers [25]. The main researches of supervisory control theory are divided into several classic problems based on Petri nets, that is, forbidden-state problems [26 , 27], forbidden-string problems [28 , 29], deadlock prevention problem [30], and liveness maintenance problems [31 , 32]. The deadlock prevention problems and liveness maintenance problems can be transformed to forbidden-state problems directly [33 , 34]. The authors in [35] address the forbidden-state problem of Petri nets by computing maximally permissive Petri net controllers. The design of control strategies for mutual exclusion mechanisms can be considered as resolving a forbidden-string problem. The authors in [36] use a colored generalized stochastic Petri net model to study the performance and correctness of a Lamport concurrent algorithm to solve the mutual exclusion problems. The authors in [37] discuss the simultaneous events in mutual exclusion transitions. The execution modes of sequential function charts are presented based on Petri nets. In [38], a method is proposed to control the occurrence of simultaneous events in mutual exclusion transitions by Petri nets based on supervisory control theory. The authors in [5] formulate two resource-sharing concepts, that is, parallel mutual exclusion and sequential mutual exclusion, and provide a theoretical basis for Petri net synthesis methods to model systems. According to these specifications, many constraints that deal with mutual exclusion problems between state and events or just events themselves can be transformed into generalized mutual exclusion constraints. This can be enforced by a set of places if all transitions are controllable. For the transitions that are uncontrollable, this specification is not always applicable. The authors in [40] provide a class of generalized mutual exclusion constraints on a class of forward-concurrent-free nets. Such a specification consists of a disjunction of conjunction of several single generalized mutual exclusion constraints. The authors in [42] enforce the generalized mutual exclusion constraints on a Petri net plant by replacing the classical partition of event set into controllable and uncontrollable events from supervisory control theory. In [43], an algorithm is presented to transform a given generalized mutual exclusion constraint into an optimal admissible one for a class of Petri nets whose uncontrollable influence subnets are forward synchronization and backward conflict-free nets. It is difficult to summarize the constraints for complex mutual exclusion mechanisms and special optimization problems in discrete event systems by using the existing control strategies, such as the following described periodic mutual exclusion problems. In many discrete event systems, one of the concurrent processes will occupy a common resource to maintain its normal running. To satisfy some special requirements, this process will consecutively obtain the common resource during the competition of the following finite times after it won a competition. The other processes only can wait for the release of the common resource. All concurrent processes will form a new competition again after this process finishes a periodical operation for the special requirements and the common resource is released. This mutually exclusive competition among these concurrent processes has periodicity, called periodic mutual exclusion behavior. It is necessary to design appropriate control strategies to distribute the common resources for these concurrent processes to control the periodic mutual exclusion problems. The structures of general mutual exclusion systems and periodic mutual exclusion systems are defined by using Petri nets. The proposal is convenient to construct formal models to analyze the properties of the two classes of mutual exclusion systems. Two original structural conversion concepts, called -derivation process and -convergence process, are proposed to construct the periodic mutual exclusion supervisors. The main idea is to derive a common resource to several virtual resources by the preliminary -derivation processes if a process obtains the common resource and these derived virtual resources will be converged into a common resource by the final -convergence processes after this process releases the common resource. The execution sequences of all processes are discussed in mutual exclusion systems with general mutual exclusion supervisors and periodic mutual exclusion supervisors, respectively. The discussion results show that many undesirable execution sequences are forbidden in the mutual exclusion systems with the periodic mutual exclusion supervisors because the periodic mutual exclusion behavior is controlled by these periodic mutual exclusion supervisors. Finally, a real example is used to illustrate the proposed methodology. The main contributions of this paper are concluded as follows: The proposal is convenient for constructing formal models for these two classes of mutual exclusion systems. The purpose is to enhance the execution permission for each process that obtains the common resource. In the enhanced mutual exclusion systems, many undesirable execution sequences of all processes are forbidden by the periodic mutual exclusion supervisors. The rest of this paper is organized as follows. Section 2 briefly recalls some basics of Petri nets. Section 3 introduces a class of mutual exclusion systems with resources and their periodic mutual exclusion behavior is discussed. Section 4 proposes the design method of periodic mutual exclusion supervisors. Section 5 gives the discussion of periodic mutual exclusion behavior in general mutual exclusion systems and periodic mutual exclusion systems. Section 6 introduces two examples to illustrate the proposed methods. Finally, the proposed methods are concluded in Section 7. Preliminaries We assume that the readers are familiar with the basics of Petri nets. Only some key concepts of Petri nets are provided. More details can be found in [44]. A Petri net is a 4-tuple , where.

Chapter 2 : Supervisory Control of Discrete Event Systems Using Petri Nets | Panos Antsaklis - calendrierc

Supervisory Control of Discrete Event Systems Using Petri Nets is intended for graduate students, advanced undergraduates, and practicing engineers who are interested in the control problems of manufacturing, communication and computer networks, chemical process plants, and other high level control applications.

Hierarchical Interface-based Supervisory Control: Wonham , " In this report we present a large manufacturing example 7: We discuss the application of our method to the Atelier Inter-etablissement de Productique AIP , a highly automated manufactu We discuss the application of our method to the Atelier Inter-etablissement de Productique AIP , a highly automated manufacturing system. We describe the system, and our supervisor design, closing by discussing the results of successfully applying our method to show that the system is nonblocking and that our supervisors are controllable. This example demonstrates that our method can be applied to interesting systems of realistic complexity that were previously far beyond our means. Show Context Citation Context A supervisor synthesis technique for Petri net plants with uncontrollable and unobservable transitions that enforces the conjunction of a set of linear inequalities on the reachable markings of the plant is presented. The approach is based on the concept of Petri net place invariants. Each step of the procedure is illustrated through a running example involving the supervision of a robotic assembly cell. The synthesis technique is based on the concept of admissible constraints. An inadmissible constraint can not be directly enforced on a plant due to the uncontrollability or unobservability of certain plant transitions. Procedures are given for identifying all admissible linear constraints for a plant with uncontrollable and unobservable transitions, as well as methods for transforming inadmissible constraints into admissib A major goal in the field of discrete event system control is the synthesis of supervisors under conditions where certain state to state transitions can not be prevented by any action from the super Synthesis of deadlock prevention supervisors using Petri nets by Marian V. When the PN is supervised so that its markings satisfy these inequalities, the supervised net is proved to be deadlockfree for all When the PN is supervised so that its markings satisfy these inequalities, the supervised net is proved to be deadlockfree for all initial markings that satisfy the supervision constraints. Deadlock-freedom implies that there will always be at least one transition that is enabled in the closed-loop supervised system. The method is not guaranteed to ensure liveness, as it can be applied to systems that cannot be made live under any circumstances. However, for controllable and observable PNs it is shown that when the method ensures liveness as well, the liveness ensuring supervisor is least restrictive. Moreover, it is shown that the method is not restrictive even for PNs in which not all transitions can be made live. The procedure allows automated synthesis of the supervisors. This paper describes an approach to the control of continuous systems through the use of symbolic models describing the system behavior only at a finite number of points in the state space. These symbolic models can be seen as abstract representations of the continuous dynamics enabling the use of These symbolic models can be seen as abstract representations of the continuous dynamics enabling the use of algorithmic controller design methods. We identify a class of linear control systems for which the loss of information incurred by working with symbolic subsystems can be compensated by feedback. We also show how to transform symbolic controllers designed for a symbolic subsystem into controllers for the original system. The resulting controllers combine symbolic controller dynamics with continuous feedback control laws and can thus be seen as hybrid systems. It provides a set of local conditions that can It provides a set of local conditions that can be used to verify global conditions such as nonblocking and controllability. As each clause of the definition can be verified using a single subsystem, the complete system model never needs to be stored in memory, offering potentially significant savings in computational resources. In this thesis, we develop a synthesis method that respects the HISC hierarchical structure. We replace the supervisor for each level by a corresponding specification DES. We then do a per level synthesis to construct for each level a maximally permissive supervisor that satisfies the corresponding HISC conditions. Abstract â€" We describe a generic routing controller for nonlinear flow shops, and the respective design and verification tool based on coloured timed Petri nets. The design methodology integrates logical and performance-related control for discrete event systems by introducing a time invariant proce The design methodology integrates logical and performance-related control for discrete event systems by introducing a time invariant process V. V stands for the cycle time of a repetitive production system and for the stationary sequence of transport and move operations. The transport system is the bottleneck resource performing as a non-linear GMEC general mutual exclusion constraint on plant behaviour. The definition of a time invariant coupling process V t is crucial for the integration of logical and performance-related control. V t stands both for the cycle Ordinary t-timed Petri Nets are used for modeling, analysis and synthesis of random topology production systems and networks. Each production system is first decomposed into production line transfer chain , assembly, disassembly and parallel machines modules and then their corresponding modular Pe Each production system is first decomposed into production line transfer chain , assembly, disassembly and parallel machines modules and then their corresponding modular Petri Net models are derived. The overall system PN model is obtained via synthesis of the generic modules satisfying system constraints. P- and T- invariants are calculated and given a random topology production system, the total number of the system PN model nodes places, transitions is calculated from the corresponding generic PN modules. Results show the applicability of the proposed methodology. In this paper we consider a high-level description of a railway network using a skeleton net that belongs to the class of ES2PR nets. The resource places of this model correspond to the action of a safeness enforcing supervisor. Liveness constraints may also be enforced for this class by adding appr Liveness constraints may also be enforced for this class by adding appropriate monitor places designed using siphon analysis. We show how this can be done without an exhaus-tive computation of all siphons and characterize the cases in which this procedure can be recursively applied, giving a simple test for the closed loop net to remain an ES2PR net. In [8] we also addressed the problem of global deadlock avoidance. In fact, when a safeness enforcing supervisor has been designed, it may well be the case that the closed loop net is not live.

DOWNLOAD PDF SUPERVISORY CONTROL OF DISCRETE EVENT SYSTEMS USING PETRI NETS

Chapter 3 : Supervisory control of discrete event systems using coloured Petri nets - CORE

Supervisory Control of Discrete Event Systems Using Petri Nets presents a novel approach to its subject. The concepts of supervisory control and discrete event systems are explained, and the background material on general Petri net theory necessary for using the book's control techniques is provided.

Hierarchical Interface-based Supervisory Control: Wonham , " In this report we present a large manufacturing example 7: We discuss the application of our method to the Atelier Inter-etablissement de Productique AIP , a highly automated manufactu We discuss the application of our method to the Atelier Inter-etablissement de Productique AIP , a highly automated manufacturing system. We describe the system, and our supervisor design, closing by discussing the results of successfully applying our method to show that the system is nonblocking and that our supervisors are controllable. This example demonstrates that our method can be applied to interesting systems of realistic complexity that were previously far beyond our means. Show Context Citation Context Although [1, 34] achieved excellent results for verifying controllability, verifying nonblocking was still a problem. This is a state based method that makes use of the algebraic regularity inherent in certain systems. It is used when the important details of the system can be expressed as a vector of integers, and Deadlock is an increasingly pressing concern as the multicore revolution forces parallel programming upon the average programmer. Existing approaches to deadlock impose onerous burdens on developers, entail high runtime performance overheads, or offer no help for unmodified legacy code. Gadara automates dynamic deadlock avoidance for conventional multithreaded programs. It employs whole-program static analysis to model programs, and Discrete Control Theory to synthesize lightweight, decentralized, highly concurrent logic that controls them at runtime. Gadara is safe, and can be applied to legacy code with modest programmer effort. Gadara is efficient because it performs expensive deadlock-avoidance computations offline rather than online. Gadara uses SBPI for control logic synthesis. In SBPI, the control synthesis problem is posed in terms of a set of linear inequalities on the marking of the Petri net. Theory and Applications , " The hybrid systems of interest contain two distinct types of components, subsystems with continuous dynamics and subsystems with discrete dynamics that interact with each other. Such hybrid systems arise in varied contexts in manufacturing, communication networks, auto-pilot design, automotive engin

Chapter 4 : supervisory control of discrete event systems using petri nets | Download eBook PDF/EPUB

by Zhonghua Zhang, W. M. Wonham - Symposium on Supervisory Control of Discrete Event Systems, This paper introduces a new synthesis approach for the supervisory control of discrete-event systems (DES).

Louis, MO, USA June , Supervisory Control of Discrete Event Systems Modeled by Mealy Automata with Nondeterministic Output Functions Toshimitsu Ushio and Shigemasa Takai Abstractâ€” In the conventional supervisory control frame- event may depend on the state at which the event occurs, and work for discrete event systems with partial event observation, may not be determined uniquely due to packet loss. State- it is assumed that, for each event, the corresponding output dependence and nondeterminism of partial event observation symbol is determined deterministically. However, this assump- tion does not hold in discrete event systems such as a system are represented by introducing a nondeterministic output with sensor errors and a mobile system, where an output symbol function that maps a pair of an event and a state into a depends on not only an event but also a state at which the event set of possible output symbols. Motivated by this, we study occurs. In this paper, we model such a discrete event system by a supervisory control of DESs modeled by Mealy automata Mealy automaton with a nondeterministic output function. We [23] with nondeterministic output functions. We present necessary and sufficient conditions control action based on a permissive policy [24] and the for each of them to achieve a given specification. We then other based on an anti-permissive one [24]. The former is present algorithms for verifying these conditions. Moreover, we called the permissive supervisor, and the latter is called the discuss the relationship between the two supervisors in the case anti-permissive supervisor. In the conventional supervisory that an output function is deterministic. We show that, however, systems DESs with partial event observation was proposed there exists a language that can be achieved only by the in [1], [2]. In this conventional framework, partial event anti-permissive supervisor under a nondeterministic output observation is represented by the projection function from the function. We introduce notions of P-observability and A- event set to the observable event set [1] or the mask function observability to characterize classes of languages achiev- from the set of events to the set of output symbols [2]. It able by the permissive supervisor and the anti-permissive was shown in [1], [2] that there exists a partial observation one, respectively. We provide effective tests to verify P- supervisor that achieves a given specification language if observability and A-observability of a given language. There are many studies dealing with supervisory achieves a given specification language. We show that A- control of DESs with partial event observation [4], [5], [6], observability is weaker than P-observability, and that con- [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17]. In trollability and A-observability are only sufficient for the ex- state feedback control of DESs, it is assumed that partial istence of a supervisor. That is, there may exist a supervisor information of the current state is available for deciding the that cannot be synthesized by using the permissive nor anti- control action [18], [19], [20]. Partial state observation is permissive policy. We then prove that in a special case that represented by the mask function from the state space to the an output function is deterministic, these sufficient conditions observation space. Further, supervisory control using partial become necessary and sufficient, and P-observability and A- event and state observations was studied [21]. In all existing observability are equivalent. In this problem, it is over a set A, the set of all prefixes of strings in K is denoted assumed that an observable event becomes unobservable after by K. For a finite failure of the corresponding sensor. Also, s denotes the symbol depends on the state of the corresponding sensor. In a mobile system, an output symbol corresponding to an II. Ushio is with the Graduate School of Engineering Science, automaton [23] with a nondeterministic output function: Osaka University, Toyonaka, Osaka , Japan, e-mail: The notation f q, s! Problem Formulation negation of f q, s!. A closed language K is said generalizes the conventional one. If there exists a supervisor S: We then have the following proposition whose proof is Fig. Controlled discrete event system G of Example 1. Permissive Supervisor We present necessary and sufficient conditions under C. Supervisory Control under the Mask Function which the permissive supervisor

SP achieves a specification We review the results on the conventional supervisory language K. We introduce a notion of P -observability, control under partial event observation. Then, the K if and only if K is controllable and P-observable. By Lemma 1, K is controllable. We prove by contradiction that K is P-observable. If K is 2 There exists a supervisor S: Then, there exists a compatible extended event Example 1: We automaton shown in Fig. Only the anti-permissive supervisor We next suppose that K is controllable and P-observable. Controlled discrete event system G of Example 2. Conversely, we assume that We next suppose that K is controllable and A-observable. We consider of [22]. We define a notion of A-observability. However, as shown in the following example, observability are necessary and sufficient conditions under these are not necessary conditions. Let K be a nonempty closed language language. The anti-permissive supervisor SA satisfies generated by an automaton shown in Fig. We suppose for contradiction that K is not A- observable. Thus, K is A-observable. Automaton representation of control specification of Example 2. In the algorithm for verifying observability [5], observable. To prove the equivalence of P-observability and are composed to characterize the violation of observability. A-observability, it suffices to show that if K is A-observable, On the other hand, we compose two copies of G and two then it is P-observable. By A- to know states reached by executing s1 and s2 in G. This observability of K, we have is a reason why we need two copies of G to verify P - observability. By Theorem 2, controllability and P-observability By Theorem 5, P-observability is verified in polynomial- are sufficient conditions for the existence of a supervisor. We prove that if there exists S: The transition function h: It can be verified that for the nonempty closed language [13] S. Then, we have the following theorem. K is A-observable if and only if [15] R. To verify A-observability using Theorem 6, we have to [16] S. We introduced two kinds of supervisors: We presented Control, vol. However, Intelligent Control Syst. Decision and Control, and European Control Conf. So the conditions presented in this System Technical J.