Chapter 103
A Formal Approach to the Distributed Software Control for Automated Multi-Axis Manufacturing Machines

Gen'ichi Yasuda
Nagasaki Institute of Applied Science, Japan

ABSTRACT

This chapter deals with the control system design problem for automated machine systems from the viewpoint of a discrete event system approach. Behaviors of automated systems have externally the features of discrete event asynchronous, concurrent processes, which implies the necessity of distributed architecture for intelligent cooperative control. Based on the generalization of multi-axis machine activities, the detailed control functions of a machine task are hierarchically represented by an interpreted form of the Petri net. The necessary control conditions and rules to ensure that the control system is well-defined, including synchronization and conflict resolution, are provided as a conceptual model of machine task. Due to hierarchical decomposition of Petri net models, the structure of the whole control system as well as the contents of each machine task is easily understood so that the task planning, monitoring, and modification of the control system can be done effectively.

INTRODUCTION

Nowadays complex automated machinery is composed of multiple independently-driven subsystems, where each subsystem is actuated by at least one effector. The evolution of an overall system is not represented by time but by events, called discrete event based state change systems as a new type of dynamic systems. Typical discrete event systems, which are essentially manmade, include computer operating systems, communication networks, traffic control systems, service systems, and manufacturing systems, especially Flexible Manufacturing Systems (FMS) and Computer Integrated Manufacturing Systems (CIMS). Modern automated intelligent machinery also has the same features as such innovative discrete event systems. In the most part of such systems, the nature of the subsystems is asynchronous and their behaviors are concurrent, independent of other subsystems. During execution of an activity of a system, each subsystem assumes its own unique state without knowledge about the state of other subsystems.

DOI: 10.4018/978-1-5225-7598-6.ch103

In such a naturally distributed hardware structure, independent axes should be synchronously activated according to events required for cooperative task execution. When activities that depend on some resources are required, the subsystems have to communicate in order to update their status and to synchronize their actions. When two activities needing a shared resource are waiting for the end of the other activity, it may be resolved according to the operational constraints with priority. Different negotiation techniques between subsystems which should cooperate can be adopted based on distributed communication structures, such as a variety of modified client/server protocols where the client and the server are not fixed, such that they are dynamically chosen according to the status of each in the system. Otherwise, a symptom of deadlock is detected and any supervisory function should be triggered based on centralized communication structures, so that, through any centralized decision making, the resource is assigned to one activity.

This chapter presents a unified and systematic design methodology of discrete event distributed control architecture by embedding intelligent agents with Petri nets. A machine-oriented, agent-based modular distributed software system is configured on a hierarchical distributed microcontroller based hardware structure for real-time intelligent machine control (Yasuda, 2010; 2011). The design of functionally distributed control software for real-time cooperative task execution is described based on the hierarchical modeling of machine tasks. Based on Petri net models of hardware elements of sensor devices such as switches, a set of program modules to execute primitive actions of effectors, is integrated with hierarchical and distributed configuration of behavior and control agents, especially to attain the synchronization of multi-axis motions involved in the complex manufacturing task.

BACKGROUND

Petri nets and automata are the most used to describe discrete event system control. Modeling and analysis of discrete event systems with controllable and uncontrollable events, turned on and off by the supervisor, were proposed based on automata theory (Ramadge & Wonham, 1987). However, finite automata present some drawbacks such as the difficulty to model parallelism, synchronization and resource sharing. Although control commands generation was related to state transitions, since the connection of several finite automata is no longer a finite automaton, the communication specifications between modules can not be achieved using the finite automata framework (Lima & Saridis, 1996; Stadter, 1999; Silva et al., 2014). In spite of a great number of researches concerning advanced methods and tools to analyze the distributed models, these models are mostly constructed by empirical methods based on the knowledge of experts and customized for a particular application. Most approaches are limited by the combinatorial explosion that occurs when attempting to model complex systems.

On the other hand, Petri nets incorporate the notion of a distributed state of a system and a rule of state change (Murata, 1989; David & Alla, 1992), providing a mathematical formalism and a graphic tool for the formal representation of a system whose dynamics is characterized by concurrency, synchronization, nondeterministic decision, mutual exclusion and conflict. Computerized automation systems have the same, typical and most important features as industrial distributed systems.

As a mathematical formalism, a matrix equation can be set up to perform structural and behavioral analysis (Wisniewski et al., 2014) using a formal state-space representation. A Petri net simulates the behaviors through the flow of tokens like a flow chart, providing a visualization of the dynamic system. Petri nets allow a modular and hierarchical synthesis approach (Gomes & Barros, 2005) which can build