Petri Net Models for Manufacturing Systems

Zdeněk Hanzálek
Department of Control Engineering, Karlovo nám. 13
Czech Technical University in Prague
121 35 Prague 2, Czech Republic
hanzalek@rtime.felk.cvut.cz

Abstract

This paper presents an experimental laboratory setup aimed at creating a flexible manufacturing system for use in teaching real-time control. The machine models (robots, conveyors, etc.) are self-contained elements with their own intelligence, communicating with a master computer. A supervisor program running on the master computer is implemented by a real-time operating system, enabling the dynamic creation of processes. The supervisor program is fully parameterized, with parameters specifying manufacturing subtasks of all machines and synchronization among them. This makes the control system modular and flexible. A manufacturing task is specified by Petri nets that are automatically decomposed into a set of unique P-invariant generators. The choice of concurrent processes corresponding to P-invariants is done semi-automatically to reflect the physical tenor of the manufacturing system.

1 Introduction

Five and half years of full-time study at the Czech Technical University is divided into three periods. The basic stage, which gives students theoretical principles of electrical engineering (namely, mathematics and physics). This stage is completed with the first state examination and its average duration is about 4 semesters.

The second period of study ends with the Bachelor's degree. A set of obligatory subjects provides basic ground for student's knowledge in a specific discipline, such as control engineering. Beside subjects oriented towards theory of control and towards electronic systems, there is a set of courses dealing with the use of computers in control engineering. These courses are supported by laboratories such as Logic Systems, Computers for Control and Operating Systems for Control. Optional courses from all-departmental offerings complete the curriculum structure with subjects from other areas. Graduates of bachelor study submit their project work and receive the title of Bachelor, after examinations.

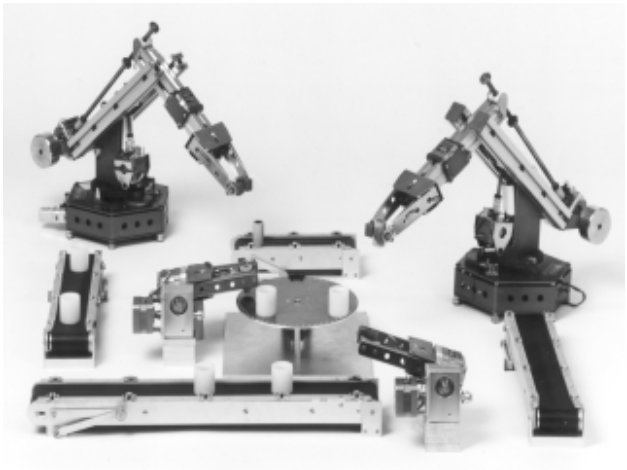

Figure 1: A flexible manufacturing system

The third and last part of full-time study at the Department of Control Engineering is to obtain the title of Enginner. In this period, students choose among many elective courses of their specialization (e.g. Control Systems, AI, Robotics, Mechatronics, Biocybernetics). Laboratories of Distributed Control, Modern Control Engineering (microcontrolers, PICs, PLCs, XLINXs) and Design of Control Systems (Intouch, Control Panel, etc.) are supporting the courses related to real-time control. At the end of the engineering period, students submit their diploma thesis and take a final state examination covering three main subjects: Theory of Control, Computers, and Electronic Systems.

Students coming to our labs are already familiar with programming (Pascal, C, low-level languages), so they can fully concentrate on conceptual aspects of systematic design. They are taught the basic principles of real-time control (real-time requirements, inter-

rupt system, synchronous/asynchronous events handling, etc.), using simple physical models (decoding incremental sensor signals, generating waveforms, etc.) in the laboratory of Computers for Control. In the two next laboratories, *Operating Systems for Control* and *Distributed Control Systems*, we use a flexible manufacturing system, whose elements are described briefly in this article.

The material related to real time is organized in a modular fashion, which allows its use in a variety of courses offered by the Department. Below, one such module is described, based on Petri nets formalism. This is followed by a presentation of an experimental setup and a summary.

2 Real-Time and Concurrency

To understand the concept of real-time requirements, students are taught the basic principles, first. In the real-time applications, a computer is connected directly to the physical equipment and is dedicated to controlling that equipment. Consequently, the system must meet response requirements that are mandated by the equipment itself, rather than those being dictated by the computer. The requirement to meet externally imposed deadlines is at the heart of what is termed a real-time system. The definition of a real-time system states that it has to respond to externally generated events within a specified and finite interval. Consequently, the software must be designed to meet these response time requirements [3].

A characteristic of a computer supervising a distributed system is that many activities within it proceed in parallel. For example, some parameters must be sampled and controlled at a very fast rate, whereas other parameters need only be sampled once per second. Logically, those two operations proceed in parallel, while of course the CPU executes them in an interleaved fashion. Similarly, controlling several I/O devices at once usually results in some parts of the control software waiting for the devices to complete an operation while other devices, having finished their operations, are being serviced.

Whether recognized or not, this parallelism, or concurrency, adds a major complication to the software. To avoid all kinds of problems, programmers must take care of mutually exclusive access to shared resources, signaling one task by another task or by an interrupt handler, and sending messages from one task or an interrupt handler to another task.

A multitasking operating system (OS) usually provides the facilities necessary to solve that kind of problems [1], [11], [12], [13]. It does it via system calls for creating and deleting tasks, suspending and resuming their execution, and so on. The system automatically takes care of task scheduling. Although tasks may logically proceed in parallel, the CPU is physically capable of running only one task at a time; therefore the scheduler interleaves their execution.

3 Petri Net Formalism

The use of Petri nets has been shown to be very promising for modelling and analysing real-time systems as well as many other concurrent systems [2]. Debugging costs are a major difficulty for real-time distributed systems. Therefore, it is valuable to perform as many checks as possible on specifications before implementation. For that reason, it is of interest to present the students with an abstract model capable to express parallelism and to derive formal proofs. Petri net (PN) is an excellent formalism of this kind for its ability to validate behavioural properties [8], [10], [14].

The state-transition dynamics in a manufacturing system are modelled as a controlled Petri net (CtlPN), which is an extension of standard Petri nets with external control inputs as additional enabling conditions on transitions [4].

Definition 1: Let a Petri net as a four-tuple $< P, T, Pre, Post >$ is such that
P is a finite and non-empty set of places
T is a finite and non-empty set of transitions
Pre is an input function (precondition)
$Post$ is an output function (postcondition).
1. A Matrix $C = (c_{ij})$ where $(1 \leq i \leq n, 1 \leq j \leq m)$ is called the incidence matrix of PN iff

$$C = Post - Pre \tag{1}$$

2. A vector $f : (1, ..., n) \in Z^+$ is called a P-invariant of the given PN, iff

$$C^T \times f = 0 \tag{2}$$

3. A vector $s : (1, ..., m) \in Z^+$ is called a T-invariant of the given PN, iff

$$C \times s = 0 \tag{3}$$

Definition 2: An invariant f of $C^T \times f = 0$ is called standardized iff f can not be written in the form $f = x_i + x_j$, where x_i, x_j are invariants and $x_i \neq 0$, $x_j \neq 0$

More generally the decomposition and the composition of a given invariant f can be described by

$$f = \sum_{i=1}^{g} \lambda_i x^i \tag{4}$$

with factors λ_i and generators x^i.

Level	$\lambda_i \in$	Generators x^i	Set $\{x^i\}$
1	Q	$x^i \in Z^n$	$\{x^i\}$ Base
2	Z	$x^i \in Z^n$	$\{x^i\}$ Base
3	Q^+	$x^i \geq 0$	$\{x^i\}$ Unique
4	Z^+	$x^i \geq 0$	$\{x^i\}$ Unique
5	{0,1}	$x^i \in \{0,1\}^n$	$\{x^i\}$ Unique

Table 1: Generator computational levels

Methods calculating invariants are published by Martinez and Silva [7] and others. Kruckenberg and Jaxy [6] considered several algorithms calculating generators and divided the computations into five levels as shown in the Table 1.

We will focus only on the third level, where each invariant is a positive linear combination of generators. It is evident that in the case of event graphs $\lambda_i \in Z^+$ and $x_i \in \{0,1\}^n$ already for the third level. The generators from this level will be called simply generators in the rest of this article. In [9], it is proved that the set of generators is finite and unique.

Kruckenberg and Jaxy give an algorithm calculating generators, based on the Kannan and Bachem [5] algorithm to calculate Hermite normal form and on the theory of polyhedral cones. This algorithm serving as a base for a task decomposition was implemented in Matlab and is available from the author upon request.

Having generators of P-invariants, it is evident which transitions have to be fired in sequence. The problem now is to choose the set of generators covering all places representing actions. The selection could be done automatically seeking for an optimal solution (e.g., minimal number of generators). But this choice is very artificial and does not reflect the physical tenor of the manufacturing system. It is better to choose the generators manually so that they represent separate machines of the manufacturing system.

To make use of this theory, students are given a manufacturing problem. Using PN simulator they first draw a Petri net model of the problem. In this environment, they simulate the behaviour of the system and extract the incidence matrix C and an initial marking vector M_0. Using Matlab, they obtain the generators of P-invariants and choose those corresponding to separate machines and serving as input data for the supervisor program described in the next section. Having the generators of P-invariants and the initial marking vector M_0, students prove that there is no deadlock in the system. In this subclass of PN called event graph, it is sufficient to prove that there is at least one token in each P-invariant. That corresponds to a simple matrix operation in Matlab.

4 Experimental Setup

The model of a technological process developed at the Department of Control Engineering, CTU Prague, is presented in Figure 1. This model consists of several kinds of machines: robots, conveyor-belts, storing plates and simple manipulators. A number of technological processes can be realized using various configurations of the machines.

Local control systems were realized by universal boards based on a microcomputer (Intel 8051) working in a multiprocessor mode. The boards are interconnected by a common link (modified serial interface RS232) with a PC acting as a master (see Figure 2).

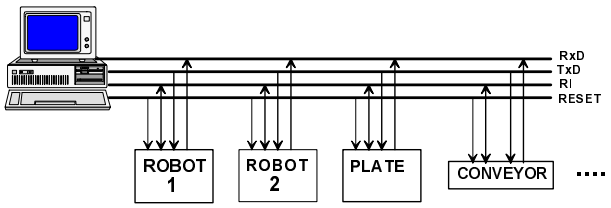

Figure 2: An experimental environment structure

The operating system kernel allows to create dynamically several concurrent processes. Figure 3 shows a block diagram of a fully parameterized program acting as a manufacturing system supervisor.

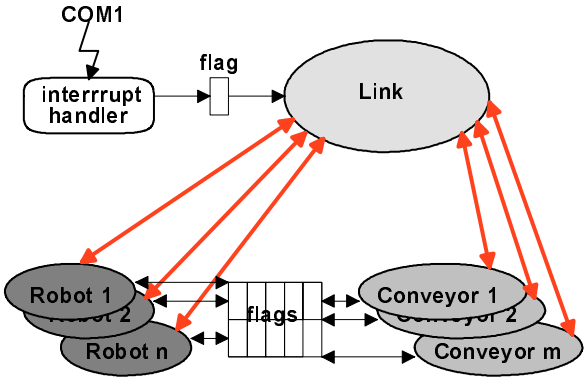

Figure 3: The supervisor structure

The program reads all the required information from a file containing specifications of separate tasks corresponding to the chosen generators. Synchronization among the generators (tasks) is done simply by

flags. The tasks are connected to machines by a common link (Figure 2) performing communication in both directions (PC → machine, machine → PC). The shared link is accessed by a special process (named Link) pending for a flag from the interrupt handler on a serial port COM1 and performing demultiplex of messages coming from the machines.

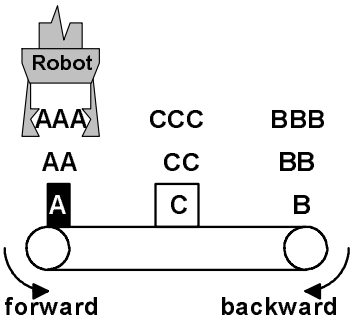

Figure 4: A sample problem

Place	Robot Activity
P_1	move to AAA
P_2	wait
P_3	move to A, close, move to AAA
P_4	wait
P_5	move to AA, open, move to AAA
P_6	move to BBB
P_7	wait
P_8	move to BB, close, move to BBB move to CCC, C, open, move to CCC
Place	**Conveyor Activity**
P_9	forward to A
P_{10}	wait
P_{11}	forward to A
P_{12}	wait
P_{13}	backward to B
P_{14}	wait
Place	**Flag Meaning**
$P_{15}...P_{20}$	wait

Table 2: The actions corresponding to places in Fig. 5

Figure 4 shows a simple manufacturing system consisting of two machines and two objects: a conveyor running forward/backward, a robot in different positions, white object in position C and black object in position A. The function of a system performing an object composition in position A and a decomposition in position B is depicted in Figure 5 and Table 2.

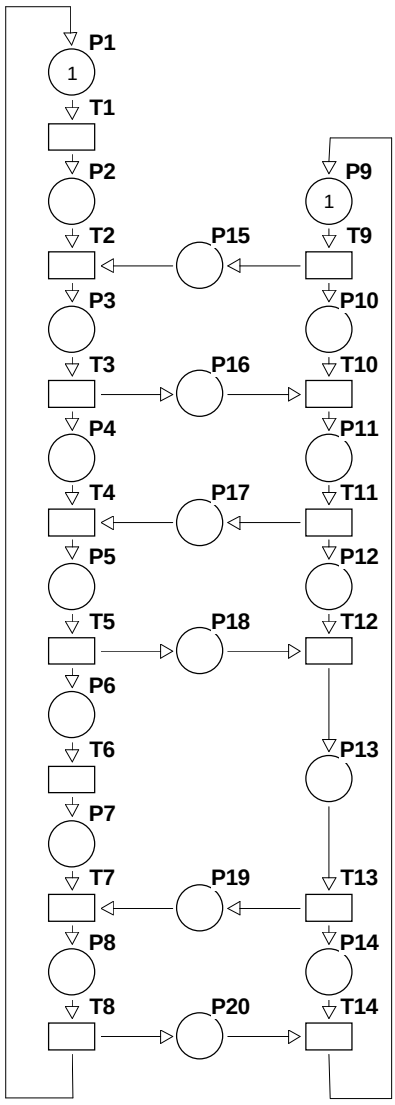

Figure 5: PN representation of a sample problem

In order to represent all the states of the system, it is necessary to guarantee that the input places to the synchronization transitions have no action meaning (e.g., P_2,P_{15} to T_2). The PN in Figure 5 can be decomposed into two P-invariants corresponding to the machines (robot $P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8$ and conveyor $P_9, P_{10}, P_{11}, P_{12}, P_{13}, P_{14}$). The two P-invariants and separate synchronization points among them ($P_{15}, P_{16}, P_{17}, P_{18}, P_{19}, P_{20}$) are given in the file

as parameters to the program acting as the supervisor. The program first creates three tasks (Robot1, Conveyor1 and Link) and six flags, and then starts the object composition and decomposition.

Handling asynchronous events with a real-time kernel is more difficult in the design phase (task decomposition is an NP-complete problem) than using a control system based on input ports polling in each sampling period. On the other hand this approach is more structural and allows more efficient programming.

The success of this procedure relies on the fact that students already have a theoretical background gained in courses on Linear Algebra and Theory of Systems. This allows them to deal with the use of the set of P-invariant generators that is unique for a given Petri net representing the manufacturing problem.

5 Summary

The system described above is used in two laboratory courses at the Department of Control Engineering. In the course *Operating Systems for Control*, students are given libraries that allow them to communicate with the manufacturing system and tune their own programs running under the multitasking OS and managing a specified manufacturing task on a fixed topology. In the course *Distributed Control Systems*, students specify various manufacturing tasks by Petri nets and decompose the problem into separate processes using algorithms that search for the set of P-invariant generators. Choosing a subset of P-invariant generators they create an input data file and run fully parameterized supervisor to control the distributed system.

The lab was realized on various software platforms (VRTX, Linux, and others). Separate implementations are used as physical models to test new components and control systems appearing on the market. New boards communicating via industrial fieldbus technologies such as CAN and Profibus, are under development. Thanks to the system modularity, it is easy to split the development into separate phases (mechanics, actuators and sensors, electronics, microcontroler programming, communication, supervisor level, PN token player, etc.) and realize them incrementally as projects done by the students.

Acknowledgments

I wish to thank anonymous reviewers, who provided comments and suggestions that improved this paper. This research has been conducted at the Department of Control Engineering as part of the research project New Control System Structures for Production Machines and has been supported by grant GACR No.102/95/0926.

References

[1] J.H.Anderson, S.Ramammurthy, K.Jeffay, "Real-Time Computing with Lock-Free Shared Objects", *Proc. Real-Time Systems Symposium*, IEEE Computer Society Press, Los Alamitos, Calif., 1995, pp. 28-38.

[2] C. Ghezzi et al., "A Unified High-Level Petri Net Formalism For Time-Critical Systems ", *IEEE Trans. Software Engineering*, 1991, pp. 160-172.

[3] Z. Hanzálek, "Real-time Neural Controller Implemented on Parallel Architecture", *in: A. Crespo (ed.): Proc. Artificial Intelligence in Real-Time Control*, Elsevier Science, Amsterdam, 1995, pp. 313-316.

[4] A. Ichikawa, K. Hiraishi, "Analysis and Control of Discrete Event Systems Represented by Petri Nets", *Discrete Event Systems: Models and Applications*, Springer-Verlag, Berlin, 1988.

[5] R. Kannan, A. Bachem, "Polynomial Algorithms for Computing the Smith and Hermite Normal Forms of an Integer Matrix", *SIAM J. Comput.*, Vol. 8, No. 4, 1979, pp. 499-507.

[6] F. Kruckeberg, M. Jaxy, "Mathematical Methods for Calculating Invariants in Petri Nets", *in: G. Rozenberg (ed.): Advances in Petri Nets*, LNCS 266, Springer-Verlag, Berlin, 1987, pp. 104-131.

[7] J. Martinez, M. Silva, "A Simple and Fast Algorithm to Obtain All Invariants of a Generalized Petri Nets", *in: C. Girault, W. Reisig (eds.): Application and Theory of Petri Nets*, Informatik Fachberichte 52, Springer-Verlag, Berlin, 1982, pp. 301-310.

[8] T.Murata, "Petri Nets: Properties, Analysis and Applications", *Proceedings of the IEEE*, Vol. 6, No. 1, 1990, pp. 39-50.

[9] K.H. Pascoletti, "Diophantische Systeme und Losungsmethoden zur Bestimmung aller Invarianten in Petri-Netzen", *Berichte der GMD*, No. 160, Bonn, 1986.

[10] J.L. Peterson, "Petri Net Theory and Modeling of Systems", Prentice Hall, Englewood Cliffs, NJ, 1981.

[11] J.A.Stankovic, K.Ramamritham, "The Spring Kernel: A New Paradigm for Real-Time Systems", *IEEE Software*, Vol. 8, No. 3, 1991, pp. 62-72.

[12] W.Tarng, T.H.Lin, "Fault-Tolerant Task Assignment in Distributed Real-Time Computing Systems", *Readings in Real-Time Systems, J. H. Lee, C.M. Krishna, eds.*, IEEE Computing Society Press, Los Alamitos, Calif., 1993, pp. 98-110.

[13] H.Tokuda, T.Nakajima, P.Rao, "Real-Time Mach: Towards a Predictable Real-Time Systems", *Proc. Usenix Mach Workshop*, 1990, pp. 1-10.

[14] R. Valette, "Analysis of Petri Nets by Stepwise Refinement", *J. Comput. Syst. Sci*, Vol. 18, 1979, pp. 35-46.