Pergamon

0005-1098(95)00103-4

Automatica, Vol. 32, No. 1, pp. 15-28, 1996
Copyright © 1995 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0005-1098/96 $15.00 + 0.00

Feedback Control of Petri Nets Based on Place Invariants*

KATERINA YAMALIDOU,† JOHN MOODY,† MICHAEL LEMMON† and PANOS ANTSAKLIS†

A computationally efficient method for constructing Petri net controllers is presented. The method is derived using place invariants and is able to enforce logical and algebraic constraints containing elements of the marking and firing vectors.

Key Words—Discrete event systems; Petri nets; invariants; supervisory control.

Abstract—This paper describes a method for constructing a Petri net controller for a discrete event system modeled by a Petri net. The controller consists only of places and arcs, and is computed based on the concept of Petri net place invariants. The size of the controller is proportional to the number of constraints that must be satisfied. This method is computationally efficient, and can accommodate constraints written as Boolean logic formulas in the conjunctive normal form or algebraic inequalities that contain elements of the marking and/or the firing vectors.

1. INTRODUCTION

Petri nets (Peterson, 1981; Reisig, 1985; Murata, 1989) are an appropriate tool for the study of discrete-event dynamical systems because of their modeling power and flexibility. In this paper a method for computing a feedback controller for a discrete-event system modeled by a Petri net is proposed. In particular, it is shown how a Petri net can be computed to restrict the behavior of a plant using a simple equation involving the plant's Petri net model and the constraints that are to be enforced. The computation involves little more than a single matrix multiplication. The computational efficiency of the method lends itself as a practical approach to controller synthesis for large and complex discrete-event systems.

Many researchers have used Petri nets as a tool for modeling, analysing and synthesizing control laws for discrete-event systems (DES). Murata *et al.* (1986) defined C nets as an extended form of safe Petri nets, and used them to construct station controllers for sequencing control with quick response time in real-time control. Boissel (1993) used simulated annealing to compute a Petri net controller for a discrete-event system modeled by a Petri net. Zhou and DiCesare (1989) proposed an adaptive design of Petri net controllers for automated manufacturing systems. They defined the controller as the control logic based on the Petri net model of the process, and used the model to generate a supervisory controller by successive augmentation. Holloway and Krogh (1990) used *controlled Petri nets* to control systems that can be modeled as cyclic controlled marked graphs, which are a special class of Petri nets. Boucher and Jafari (1992) have presented a method of transforming controller designs using the 'structured analysis and design technique' and the 'integrated computer manufacturing definition 0' into Petri nets in order to take advantage of their graphical and computational efficiencies. Giua and DiCesare (1994) have done work on language control and realization with Petri nets, extending some of the control synthesis ideas for automata developed by Ramadge (1989) and Wonham (1987). An informative study of Petri net control issues can be found in Holloway and Krogh (1994).

The control method presented in this paper uses place invariants, a structural Petri net property, to compute a feedback controller. Valette *et al.* (1985) and Valette (1986) have explored the use of Petri nets as analytical and computational models for the representation of

* Received 9 May 1994; revised 17 November 1994; received in final form 5 May 1995. This paper was not presented at any IFAC meeting. This paper was recommended for publication in revised form by Associate Editor Peter Fleming under the direction of Editor Yaman Arkun. Corresponding author Dr John Moody. Tel. +1 219 631 5480/5792; Fax +1 219 631 5480; E-mail antsakli@saturn.ee.nd.edu.

† Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556, U.S.A.

discrete event system controllers, and have used Petri net place invariants as an analysis tool. In this paper invariants are not used to analyze, but rather to actually synthesize controls that enforce linear constraints on the marking behavior of the Petri net that is to be controlled. The constraints handled by the proposed method are the same type of constraints discussed by Li and Wonham (1993, 1994) in their recent work on Petri net control, where they concentrate on controllability issues and discuss other problems such as formal language realization and dynamic state feedback. The method they present for controller synthesis involves solving a set of linear integer programming problems. The emphasis of this paper is on the efficient computation of Petri net controllers given an appropriate set of constraints rather than the actual controllability of the plant or feasibility of the constraints.

In order to compute a Petri net controller using place invariants, it is necessary that the constraints be linear inequalities composed of elements of the Petri net marking vector. Fortunately, it is possible to transform many other constraints on a plant's behavior into such inequalities. Yamalidou and Kantor (1991) have shown how constraints written as Boolean expressions can be transformed into sets of linear inequalities involving the firing and marking vectors. It is shown here how constraints that involve the firing vector can be transformed, using two different methods, into constraints that involve only making vector elements. Thus the control method can be applied to systems whose constraints can be expressed as inequalities, inequalities or logic expressions, and may involve elements of the marking and/or the firing vector.

The controller derived using the approach in this paper is maximally permissive in that it forces the set of constraints to be obyed, while allowing any action that is not directly or indirectly forbidden by the constraints. If the constraints on a net's performance are written in terms of the firing vector then there are situations in which the maximal permissiveness of the controller can only be guaranteed if the net is safe. The control method is general and also computationally very efficient, since it involves little more than a matrix multiplication. For this reason, feedback controllers can be derived for very large Petri net models, opening the way to the design of feedback control systems for large and complex industrial applications.

The paper is structured as follows. The control design method is presented in Section 2, along with a brief discussion of Petri net place invariants and some examples. Different kinds of constraints and constraint transformations are discussed in Section 3. An example is presented in Section 4 that includes constraint transformation as well as controller computation. Concluding remarks are given in Section 5.

2. DESCRIPTION OF THE METHOD

The system to be controlled is modeled by a Petri net with n places and $\dot{m}$ transitions, and is known as the plant or *process net*. The incidence matrix of the process net is D_p. It is assumed that all the enabled transitions can fire. It is possible that the process net will violate certain constraints placed on its behavior—hence there is a need for control. The *controller net* is a Petri net with incidence matrix D_c made up of the process net's transitions and a separate set of places. The *controlled system* or *controlled net* is the Petri net with incidence matrix D made up of both the original process net and the added controller. The control goal is to force the process to obey constraints of the form

$$\sum_{i=1}^{n} l_i\mu_i \le \beta, \tag{1}$$

where μ_i is the marking of place p_i, and the l_i and β are integer constants. For example, we might wish to enforce the constraint $\mu_1 + \mu_2 \le 1$, which means that at most one of the two places p_1 and p_2 can be marked, or, in other words, both places cannot be marked at the same time. This inequality constraint can be transformed into an equality by introducing a nonnegative *slack variable* μ_c into it. The constraint then becomes $\mu_1 + \mu_2 + \mu_c = 1$, or, in general,

$$\sum_{i=1}^{n} l_i\mu_i + \mu_c = \beta. \tag{2}$$

The slack variable in this case represents a new place p_c, which holds the extra tokens required to meet the equality. It ensures that the weighted sum of tokens in the process net's places is always less than or equal to β. The place that maintains the inequality constraint is part of a separate net called the controller net. The structure of the controller net will be computed by observing that the introduction of the slack variable introduces a place invariant for the overall controlled system defined by (2). A brief review of Petri net place invariants is given in Section 2.1, before the actual controller computation is presented in Section 2.2. The maximal permissiveness of the control method is examined in Section 2.3, and the 'cat-and-mouse' problem of DES literature is used as an example of controller construction in Section 2.4.

2.1. *Place invariants*

One of the structural properties of Petri nets, i.e. properties that depend only on the topological structure of the Petri net and not on the net's initial marking, are the net invariants. Here we are interested in *place invariants*; Petri nets may also contain transition invariants (Reisig, 1985; Murata, 1989). Invariants are important means for analyzing Petri nets, since they allow the net's structure to be investigated independently of any dynamic process (Lautenbach, 1987).

Place invariants are sets of places whose token count remains constant for all possible markings. A single invariant is represented by an n-column vector x, where n is the number of places of the Petri net, whose nonzero entries correspond to the places that belong to the particular invariant and zeros everywhere else. A place invariant is defined as every integer vector x that satisfies

$$x^T\mu = x^T\mu_0, \tag{3}$$

where μ_0 is the net's initial marking, and μ represents any subsequent marking. Equation (3) means that the weighted sum of the tokens in the places of the invariant remains constant at all markings, and this sum is determined by the initial marking of the Petri net. The place invariants of a net can be computed by finding the integer solutions to

$$x^T D = 0, \tag{4}$$

where D is the $n \times m$ incidence matrix of the Petri net, with n being the number of places and m the number of transitions of the net. It is easily shown that every linear combination of place invariants is also a place invariant for the net. Below, it is described how place invariants can be used to simply compute a controller for a plant modeled by a Petri net.

2.2. *Controller computation*

Each constraint of the type (1) enforced on the net will have a slack variable associated with it, and each slack variable will be represented in the controller net as a place. Thus the size (number of places) of the controller net is proportional to the number of constraints that are to be enforced. Every place used to control the process nets adds one row to the incidence matrix D of the controlled system. Thus D is composed of two matrices, the original $n \times m$ matrix D_p of the process model and the incidence matrix of the controller, called D_c. The arcs connecting the controller place to the original Petri net of the system will be computed by the place-invariant equation (4), where the unknowns are the elements of the new row of the matrix D and the vector x is the desired place invariant defined by (2), i.e. $x^T = [l_1 \; l_2 \; \ldots \; l_n \; 1]$.

The control problem can be stated in general as follows. All constraints of the type (1) can be grouped and written in matrix form as

$$L\mu_p \le b, \tag{5}$$

where μ_p is the marking vector of the Petri net modeling the process, L is an $n_c \times n$ integer matrix, b is an $n_c \times 1$ integer vector and n_c is the number of constraints of the type (1). Note that the inequality is with respect to the individual elements of the two vectors $L\mu_p$ and b, and can be thought of as the logical conjunction of the individual 'less than or equal to' constraints. All place-invariant equations of the type (2), generated after the introduction of the slack variables, can be grouped in matrix form as follows:

$$L\mu_p + \mu_c = b, \tag{6}$$

where μ_c is an $n_c \times 1$ integer vector that represents the marking of the controller places.

Each place invariant defined by (6) must satisfy (4):

$$X^T D = [L \quad I]\begin{bmatrix} D_p \\ D_c \end{bmatrix} = 0,$$

$$LD_p + D_c = 0,$$

where X is a matrix representing the n_c different invariants and I is an $n_c \times n_c$ identity matrix, since the coefficients of the slack variables in the constraints are all equal to 1. The matrix D_c contains the arcs that connect the controller places to the transitions of the process net. So, given the Petri net model of the process (D_p) and the constraints that the process must satisfy (L and b), the Petri net controller D_c is defined by

$$D_c = -LD_p. \tag{7}$$

Since our method admits the structure of the process net as well as a set of specifications, it can control transitions that participate in self-loops in the process net. This is because the constraints on these transitions are part of the specifications. Note that when an element of D_c is zero, there are no arcs at all connecting the given place and transition, i.e. there are no cancelling self-loops in the controller structure. Self-loops may occur if the graph transformation techniques discussed in Section 3.5 are used.

The initial marking of the controller Petri net μ_{c_0} is calculated so that the place-invariant equation (6) is initially satisfied. Given (3), (6) can be written for the initial marking vector

$$L\mu_{p_0} + \mu_{c_0} = b.$$

Therefore

$$\mu_{c_0} = b - L\mu_{p_0}. \tag{8}$$

Example. Consider the simple Petri net of Fig. 1, which is acyclic and nonsafe. Its incidence matrix is

$$D_p = \begin{bmatrix} -1 & 0 & 0 & 1 & 0 \\ 1 & 1 & -1 & 0 & 1 \\ 0 & -1 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix},$$

while its initial marking is

$$\mu_{p_0} = [\mu_1 \quad \mu_2 \quad \mu_3 \quad \mu_4]^T = [3 \quad 0 \quad 0 \quad 3]^T$$

D_p is rank 3; thus it has one place invariant that includes the entire net, i.e. $x^T D_p = 0$, where $x^T = [1\ 1\ 1\ 1]$. The objective is to control the net so that places p_2 and p_3 never contain more than one token; i.e. we wish to enforce the constraint

$$\mu_2 + \mu_3 \leq 1. \tag{9}$$

Using the matrix notation of (5), we have

$$L = [0 \quad 1 \quad 1 \quad 0],$$

$$b = 1.$$

The process net will not satisfy the desired constraint without an external controller. A slack variable μ_c is introduced, and the inequality (9) becomes an equality:

$$\mu_2 + \mu_3 + \mu_c = 1. \tag{10}$$

The slack variable μ_c denotes the marking of the place p_c that belongs to the controller. Equation (10) represents the desired invariant $X^T = [0\ 1\ 1\ 0\ 1]$, which will be forced on the controller system. The incidence matrix of the controller net is computed using (7);

$$D_c = -LD_p = [-1 \quad 0 \quad 0 \quad 1 \quad -1].$$

The initial marking of the controller place is computed from (8):

$$\mu_{c_0} = -L\mu_{p_0} = 1.$$

The Petri net graph of the controlled system is shown in Fig. 2. The controller arcs are shown with dashed lines, and the control place is drawn thicker than the process places.

2.3. *Maximal permissiveness*

The control method can be shown to be maximally permissive by examining the place invariants of the controlled system. Let X_p be an integer matrix of linearly independent columns representing a basis for the place invariants of the (uncontrolled) process net. Then X_p satisfies the equation

$$X_p^T D_p = 0,$$

where the columns of X_p are linearly independent, and the number of columns of X_p (and thus the number of invariants) is equal to $n - \operatorname{rank} D_p$, since D_p is an $n \times m$ matrix and X_p forms a basis for the null space of D_p. Note that if $\operatorname{rank} D_p = n$ then the uncontrolled plant has no place invariants.

A controller is constructed using (7). The incidence matrix of the controlled net is then

$$D = \begin{bmatrix} D_p \\ D_c \end{bmatrix} = \begin{bmatrix} D_p \\ -LD_p \end{bmatrix}.$$

Note that, since the rows of D_c are linear combinations of the rows of D_p, $\operatorname{rank} D = \operatorname{rank} D_p$. Thus the number of invariants of the controlled system is equal to $n + n_c - \operatorname{rank} D_p$. All of these invariants are accounted for by the uncontrolled plant invariants, and the forced

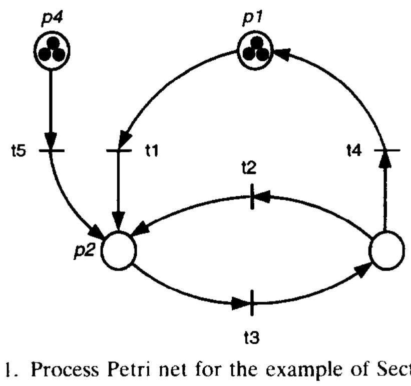

Fig. 1. Process Petri net for the example of Section 2.2.

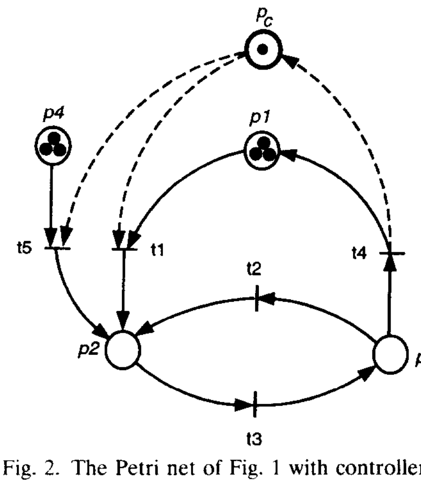

Fig. 2. The Petri net of Fig. 1 with controller.

constraint invariants as shown below. First note that

$$[X_p^T \quad 0]\begin{bmatrix} D_p \\ D_c \end{bmatrix} = X_p^T D_p = 0.$$

Thus the invariants of the uncontrolled plant are also invariants of the controlled plant. This is true for any Petri net control scheme that only adds places and arcs in order to control the plant. From the construction of the control law, we also know that

$$[L \quad I]\begin{bmatrix} D_p \\ D_c \end{bmatrix} = LD_p + D_c = LD_p - LD_p = 0,$$

and thus all $n + n_c - \text{rank}\, D_p$ invariants of the controlled net are given by $X_c^T D = 0$, where

$$X_c = \begin{bmatrix} X_p & L \\ 0 & I \end{bmatrix}.$$

The rank (and number of columns) of X_c is $n + n_c - \text{rank}\, D_p$, since X_p is $\text{rank}\, n - \text{rank}\, D_p$ and I is an $n_c \times n_c$ identity matrix.

There are no new or unexpected invariants forced on the system as a result of the control law. The control law is maximally permissive, since no action is prohibited that is not a result of the plant structure itself or the constraints forced on the plant.

2.4. *Example*: *the cat-and-mouse problem*

The 'cat-and-mouse' problem, introduced by Wonham and Ramadge (1987), is a popular example in the field of discrete event system control. The problem involves a maze of five rooms where a cat and a mouse can circulate. The rooms are connected with doors through which the animals can pass, as shown in Fig. 3.

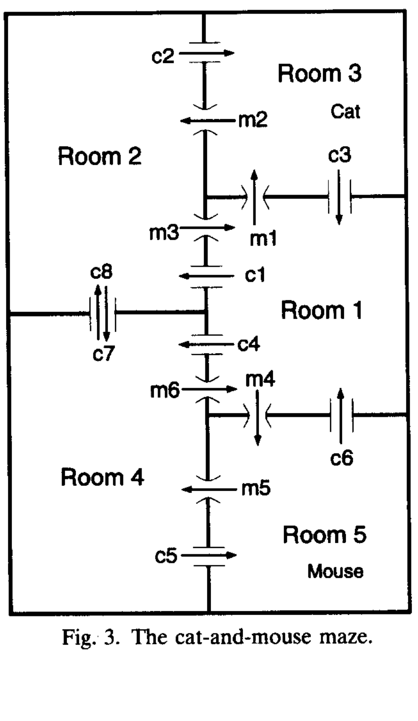

Fig. 3. The cat-and-mouse maze.

The problem is to control the doors so that the cat and the mouse can never be in the same room at the same time. The controller should be maximally permissive in the sense that it should grant maximum freedom of movement to both the cat and the mouse. The simple Petri net model of the cat-and-mouse problem is taken from Boissel (1993) and is shown in Fig. 4. The upper net concerns the cat, while the lower net concerns the mouse. Each net has five places, which model the five rooms of the maze. The transitions model the ability of each animal to pass from one room to the other according to Fig. 3.

The incidence matrix of the Petri net model for this system is

$$D_p = \begin{bmatrix}
-1 & 0 & 1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & -1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0
\end{bmatrix}.$$

There is one token only in each of the subnets, since there is one cat and one mouse. The presence of a token in a place indicates that the animal modeled by the token is in the room modeled by the particular place. Initially the cat is in room 3 and the mouse is in room 5, so the initial marking is

$$\begin{aligned} \mu_{p_0} &= [\mu_1 \quad \mu_2 \quad \cdots \quad \mu_{10}]^T \\ &= [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1]^T. \end{aligned}$$

D_p is rank 8; this it has two place invariants. The first consists of all five places of the upper

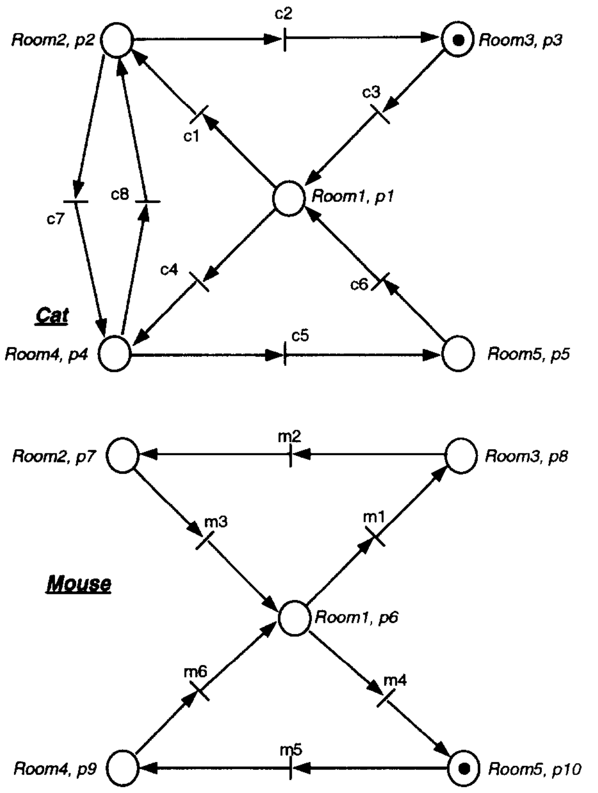

Fig. 4. The Petri net model of the cat-and-mouse problem.

subnet, and the second consists of all five places of the lower subnet.

Assume that all the doors of the maze are controllable. The control goal is to ensure that the cat and the mouse are never in the same room simultaneously. This means that each pair of places, one from the upper net and one from the lower net, that model the same room must never contain more than one token. This requirement is translated into the following five constraints:

$$\begin{aligned}&\mu_1+\mu_6\le 1,\quad \mu_2+\mu_7\le 1,\quad \mu_3+\mu_9\le 1,\\ &\mu_4+\mu_9\le 1,\quad \mu_5+\mu_{10}\le 1.\end{aligned}\tag{11}$$

The constraints are forced into equalities by introducing slack variables

$$\begin{aligned}&\mu_1+\mu_6+\mu_{c_1}=1,\quad \mu_2+\mu_7+\mu_{c_2}=1,\\ &\mu_3+\mu_8+\mu_{c_3}=1,\quad \mu_4+\mu_9+\mu_{c_4}=1,\\ &\mu_5+\mu_{10}+\mu_{c_5}=1.\end{aligned}\tag{12}$$

The five slack variables correspond to five new places, which belong to the controller, i.e. $\mu_c=[\mu_{c_1}\ \mu_{c_2}\ \mu_{c_3}\ \mu_{c_4}\ \mu_{c_5}]^T$.

Using the matrix notation of (6), we have

$$L=\begin{bmatrix}1&0&0&0&0&1&0&0&0&0\\0&1&0&0&0&0&1&0&0&0\\0&0&1&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&0&1&0\\0&0&0&0&1&0&0&0&0&1\end{bmatrix},\quad b=\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix}.$$

The portion of the matrix D associated with the controller is given by (7).

$$D_c=-LD_p$$

$$=\left[\begin{array}{cccccccc}1&0&-1&1&0&-1&0&0\\-1&1&0&0&0&0&1&-1\\0&-1&1&0&0&0&0&0\\0&0&0&-1&1&0&-1&1\\0&0&0&0&-1&0&0&0\end{array}\right.$$
$$\left.\begin{array}{cccccc}1&0&-1&1&0&-1\\0&-1&1&0&0&0\\-1&1&0&0&0&0\\0&0&0&0&-1&1\\0&0&0&-1&1&0\end{array}\right].\tag{13}$$

The initial marking of the slack places is given by (8).

$$\mu_{c_0}=1-L\mu_{p_0}=[1\quad 1\quad 0\quad 1\quad 0]^T.$$

The incidence matrix and initial marking of the controlled system are

$$D=\begin{bmatrix}D_p\\D_c\end{bmatrix},\quad \mu_0=\begin{bmatrix}\mu_{p_0}\\\mu_{c_0}\end{bmatrix}.$$

The new net prevents the cat and mouse from evern entering the same room, but allows all other legal moves. The controlled cat-and-mouse system is shown in Fig. 5. Once again, dashed lines and thick-lined places are used to represent the elements of the controller Petri net.

The 'cat-and-mouse' problem presented in Wonham and Ramadge (1987) specifies that transitions c_7 and c_8 are uncontrollable. This was not considered in the example above, because the focus of this paper is not on controllability but on efficient controller computation *given an appropriate set of constraints.* It is posssible to specify constraints in the form (5) to realize the solution of this problem even with the uncontrollable transitions; however, the constraints themselves are not as straightforward as those used above. This issue is briefiy discussed below.

Suppose we wish to write an appropriate set of constraints such that the cat and the mouse never meet and so that each animal is able to find its way back to its original room. This goal must be met while accounting for all valid initial conditions and the uncontrollability of transitions c_7 and c_8. First note that, because of the uncontrollability, the cat cannot be prevented from moving back and forth between rooms 2 and 4. So at most one animal should be allowed in rooms 2 and 4 at any time. Now suppose that initially the mouse is in room 4 and the cat is still at its normal starting position in room 3. If the cat is allowed to move to room 1, both animals

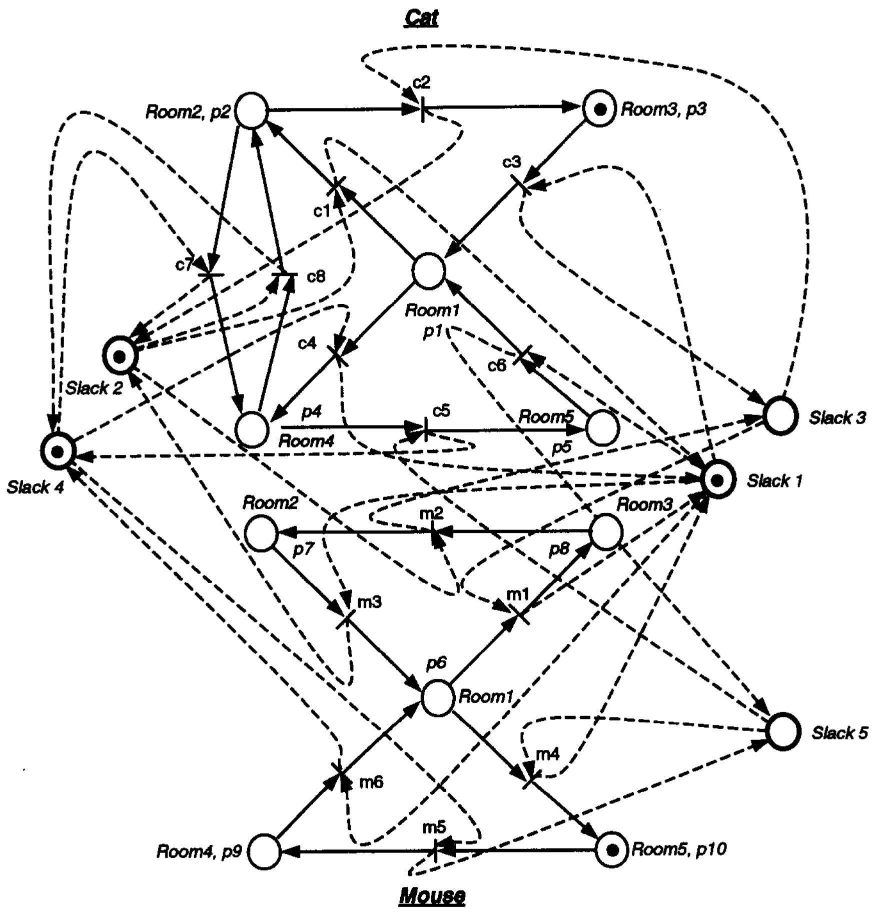

Fig. 5. The cat-and-mouse Petri net and its Petri net controller.

will be stuck, because the cat cannot be allowed to go into room 2 since it could then catch the mouse in room 4, and the only exit door from room 4 for the mouse is into the cat's current position in room 1. Thus, if we are to be certain that the animals can return to their starting rooms, we must make sure that no more than one animal inhabits rooms 1, 2 and 4 at any given time. In order to complete that set of constraints for this example, we include the remaining two constraints in (11) concerning rooms 3 and 5 respectively, to obtain

$$\begin{aligned} \mu_1 + \mu_2 + \mu_4 + \mu_6 + \mu_7 + \mu_9 &\le 1, \\ \mu_3 + \mu_8 &\le 1, \qquad (14) \\ \mu_5 + \mu_{10} &\le 1. \end{aligned}$$

At this point, the controller can be computed using the method shown above (Yamalidou *et al.*, 1994).

3. CONSTRAINT TRANSFORMATIONS

Since the method presented here is based on the concept of place invariants, the constraints must be expressed as a weighted sum of μs so that the controller can be constructed as discussed in Section 2. However, not all constraints are initially written in this form, so all other types of constraints must be first transformed to the form that the method can handle. This section examines diffeent types of constraints, and describes approriate transformations for each of them, based on the concept of replacing an element of the firing vector q with the sum of the input places of the corresponding transition. Three general categories are distinguished: logical constraints written as Boolean logic formulas in the conjunctive normal form, inequality constraints and equality constraints. Several cases are considered within each category. In addition, a graphical transformation method is briefly described.

3.1. *Logical constraints*

If a constraint can be written as a well-formed Boolean formula in the conjuctive normal form

$$A \rightarrow \Phi_1 \wedge \Phi_2 \wedge \ldots \wedge \Phi_g, \qquad (15)$$

where A is a propositional variable and each of the Φ_i is an elementary disjunction of the form

$$\Phi_i \equiv \Psi_{i_1} \vee \Psi_{i_2} \vee \ldots \vee \Psi_{i_{h_i}}, \qquad (16)$$

and each of the Ψ_{i_j} is a propositional variable, then it has been proved (Yamalidou and Kantor, 1991) that it can be translated to an equivalent set of g simultaneous linear inequalities

$$(1-\Psi_{i_1})+(1-\Psi_{i_2})+\ldots+(1-\Psi_{i_{h_i}})+A \leq h_i \quad \text{for } i=1,2,\ldots,g. \tag{17}$$

The set of all inequalities (17) will now represent the constraint that the system must satisfy. The system of inequalities should be refined after this transformation in order to eliminate those inequalities that appear more than once. The inequalities produced from the transformation of (15) can then be handled as described in the approriate sections below.

3.2. *'Less than or equal to' constraints*

3.2.1. *Constraints containing marking vector elements only.* Assume that the constraints that must be satisfied by the system can be written as a sum of elements of the marking vector:

$$\sum_{i=1}^{r} \mu_i \leq k. \tag{18}$$

This means that the sum of tokens in places $p_1, p_2, \ldots, p_r$ of the Petri net should never exceed the integer k. This type of constraint is already in the desired form, since it is an inequality similar to that shown in (1). By introducing a slack variable μ_c, it can be forced to become an equality:

$$\sum_{i=1}^{r} \mu_i + \mu_c = k. \tag{19}$$

The method then applies as described in Section 2. Constraints that contain a weighted sum of μs,

$$\sum_{i=1}^{r} l_i \mu_i \leq k, \tag{20}$$

are treated in the same way. The integer coefficients l_i will be contained in L.

Note that the constant k in (18) and (20) does not play any role in the structure (i.e. the number of places or arcs) of the controller. It defines the constant token count of the invariant described by (19), and should be taken into account when the Petri net is initially marked.

3.2.2. *Constraints containing both marking and firing vector elements.* Such constraints link the occurrence of events to part or all of the current state vector, and they denote the actions allowed by the system's state. Since the method described in this paper applies to expressions containing elements of the marking vector only, constraints that contain elements of both the marking and the firing vectors must be transformed to an appropriate equivalent form. Below, we show how this is done, and we distinguish various cases. Note that these transformations produce a maximally permissive controller for *safe* Petri nets only, i.e. for nets whose places can receive at the most one token.

Consider constraints that contain only one element of the firing vector and have the form

$$\sum_{i=1}^{r} \mu_i + q_j \leq r, \tag{21}$$

where the number of μs involved in the constraint is equal to the constant of the right side r. This constraint means that t_j cannot fire if all the places $p_1, p_2, \ldots, p_r$ are marked. Using the enabling condition of Petri net theory, which states that a transition can fire if and only if all its input places are marked, we may replace q_j in (21) with the sum of its input places and modify the right part of the inequality accordingly. The transformed constraint becomes

$$\sum_{i=1}^{r} \mu_i + \sum_{j=1}^{c_j} \mu_j \leq r + c_j - 1, \tag{22}$$

where c_j is the number of input places of t_j. The constraint in its new form will not allow more than $r + c_j - 1$ of the places in the left-hand side of (22) to be marked. So, if all the places $p_1, p_2, \ldots, p_r$ are marked then not all the input places of transition t_j may be marked, and consequently t_j cannot fire. On the other hand, if all the input places of t_j are marked then at the most $r-1$ of the places $p_1, p_2, \ldots, p_r$ can be marked. The constraint in its new form (22) contains only elements of the marking vector. Note that some of the input places of q_j may be involved in the sum of marking elements in (21). This is taken into account in (22), since they are counted twice in the right side, in both r and c_j.

If the constraint has the form

$$\sum_{i=1}^{r} \mu_i + q_j \leq k, \tag{23}$$

where $k < r$, then two cases must be distinguished. If the transition t_j has only one input place, p_j, then q_j in the above expression can be replaced by the marking μ_j of that input place, and the constraint becomes

$$\sum_{i=1}^{r} \mu_i + \mu_j \geq k. \tag{24}$$

The transformed constraint (24) is equivalent to (23), but contains only marking vector elements.

If the transition t_j has more than one input place then the constraint (23) must first be replaced by an equivalent set of constraints that list all the cases not allowed by (23), each of which is in a form that can be transformed to (18). This set will have the form

Part 1 $\sum_{i=1}^{r} \mu_i \leq k,$

Part 2

$$\begin{cases} \mu_1 + \mu_2 + \ldots + \mu_{k-1} + \mu_k + q_j \leq k, \\ \mu_1 + \mu_2 + \ldots + \mu_{k-1} + \mu_{k+1} + q_j \leq k, \\ \vdots \\ \mu_1 + \mu_2 + \ldots + \mu_{k-1} + \mu_r + q_j \leq k, \\ \mu_2 + \mu_2 + \ldots + \mu_k + \mu_{k+1} + q_j \leq k, \\ \vdots \\ \mu_2 + \mu_3 + \ldots + \mu_k + \mu_r + q_j \leq k, \\ \vdots \\ \mu_{r-k+1} + \mu_{r-k+2} + \ldots + \mu_{r-1} + \mu_r + q_j \leq k. \end{cases} \tag{25}$$

The inequality in the first line of (25) is already in the form of (18). The second line contains as many inequalities as are the combinations of r elements taken by k. Each of these inequalities involves k marking elements from the original r of (23) and q_j, and is in the form of (21), so it can be further transformed as shown above.

Lastly, the constraint may have the more general form

$$\sum_{i=1}^{r} \mu_i + \sum_{j=1}^{g} q_j \leq k, \tag{26}$$

where $k < r + g$. This type of constraint must be first replaced by an equivalent set of constraints, i.e. a set of inequalities that describe all the forbidden situations contained in (26), each of which will be in one of the forms described in this section and will contain at the most one marking element q_j, $j = 1, \ldots, g$. Each can then be transformed appropriately, as shown elsewhere in this subsection.

3.2.3. *Constraints involving firing vector elements only.* The constraints refer to the simultaneous occurrence of two or more events. All the possible cases are considered. The transformations are valid for safe nets only.

Assume that the constraint is of the type

$$\sum_{j=1}^{r} q_j \leq r - 1, \tag{27}$$

which indicates that not all transitions $q_1, q_2, \ldots, q_r$ can fire simultaneously. The enabling condition of Petri net theory suggests that the constraint can be transformed into an equivalent constraint by replacing each transition with the sum of its input places in the left-hand side of the inequality (27) and by modifying the right-hand side appropriately. The constraint then becomes

$$\sum_{j=1}^{r} \mu_{j_1} + \mu_{j_2} + \ldots + \mu_{c_j} \leq \sum_{j=1}^{r} c_j - 1, \tag{28}$$

where $\mu_{j_1}, \mu_{j_2}, \ldots, \mu_{c_j}$ are the input places of the transition t_j, while c_j denotes their number. The transformed constraint does not allow all of the input places of all transitions to be marked simultaneously, so it ensures that not all transitions can fire together and it contains only elements of the marking vector.

Any other type of constraint containing only elements of the firing vector has the form

$$\sum_{j=1}^{r} q_j \leq k, \tag{29}$$

where $k \leq r - 2$. This must be first replaced by an equivalent set of inequalities of the type (27), each of which will contain $k + 1$ firing vector elements:

$$\begin{gathered} q_1 + q_2 + \ldots + q_k + q_{k+1} \leq k, \\ q_1 + q_2 + \ldots + q_k + q_{k+2} \leq k, \\ \vdots \\ q_1 + q_2 + \ldots + q_k + q_r \leq k, \\ q_2 + q_3 + \ldots + q_{k+1} + q_{k+2} \leq k, \\ \vdots \\ q_2 + q_3 + \ldots + q_{k+1} + q_r \leq k, \\ \vdots \\ q_{r-k} + q_{r-k+1} + \ldots + q_{r-1} + q_r \leq k. \end{gathered} \tag{30}$$

Each inequality in the above set is in the form (27), and can be transformed as shown earlier.

3.3. *'Greater than or equal to' constraints*

3.3.1. *Constraints containing marking vector elements only.* In some cases it is necessary to say that a set of places contains at least k tokens. This is expressed by the constraint

$$\sum_{i=1}^{r} \mu_i \geq k. \tag{31}$$

This means that at least k of the r places must be marked. This constraint can become an equality if we add an *excess* variable μ_e to its left-hand side:

$$\sum_{i=1}^{r} \mu_i - \mu_e = k. \tag{32}$$

As in the case of the slack variable, the excess variable μ_e introduces a place that belongs to the controller and forces an invariant formed by places $\mu_1, \mu_2, \ldots, \mu_r$ and μ_e in the controlled Petri net. As shown by (32), it is the *weighted* sum of the tokens in the places of the invariant that remains constant. Since the constraint has been brought into the form of an invariant, it can be treated similarly to (19), the only difference being that the coefficients of the slack variables are now -1. The structure of the controller net is computed as

$$[L \quad -I]\begin{bmatrix} D_p \\ D_c \end{bmatrix} = 0,$$

$$LD_p - D_c = 0;$$

therefore

$$D_c = LD_p, \tag{33}$$

while the initial vector of the controller places is

$$\begin{aligned} L\mu_{p_0} - \mu_{c_0} &= b \\ \mu_{c_0} &= L\mu_{p_0} - g. \end{aligned} \tag{34}$$

3.3.2. *Constraints containing both marking and firing vector elements.* The transformations described in this section are valid for safe Petri nets only.

The first case considered is the constraint containing one element of μ and one element of q:

$$\mu_i + q_j \geq 1. \tag{35}$$

This expression means that whenever μ_i is not marked, t_j should fire, and whenever t_j does not fire, μ_i should be marked. In order to transform this constraint to an expression containing elements of the marking vector only, we must first express the constraint as a well-formed Boolean formula. It can be replaced by the following logical expression, since they are logically equivalent:

$$\neg\mu_i \rightarrow q_j. \tag{36}$$

This simply means that whenever μ_i is not true (i.e. p_i is not marked), q_j must be true (i.e. t_j must fire), and whenever q_j is not true (t_j does not fire), μ_i must be true (i.e. p_i must be marked). According to Petri net theory, a transition can fire if and only if all its input places are marked. This allows us to replace q_j in (36) with the conjunction of all its input places. It then becomes

$$\neg\mu_i \rightarrow \mu_{j_1} \wedge \mu_{j_2} \wedge \ldots \wedge \mu_{c_j}, \tag{37}$$

where c_j is the number of input places of t_j. This contains only elements of the marking vector. It is a well-formed formula of the type (15) and, as shown in Section 3.1, it is equivalent to and can be replaced by the following set of inequalities:

$$\begin{aligned} &(1-\mu_{j_1}) + (1-\mu_i) \leq 1 \Rightarrow \mu_{j_1} + \mu_i \geq 1, \\ &(1-\mu_{j_2}) + (1-\mu_i) \leq 1 \Rightarrow \mu_{j_2} + \mu_i \geq 1, \\ &\vdots \\ &(1-\mu_{c_j}) + (1-\mu_i) \leq 1 \Rightarrow \mu_{c_j} + \mu_i \geq 1. \end{aligned} \tag{38}$$

The term $\neg\mu_i$ has been replaced by $1-\mu_i$, because the net is safe. This substitution holds, because $\neg\mu_i$ and $1-\mu_i$ have the same value:

$$\begin{aligned} &\text{for } \mu_i = 1 \quad \neg\mu_i \equiv 0 \text{ and } 1-\mu_i = 0, \\ &\text{for } \mu_i = 0 \quad \neg\mu_i \equiv 1 \text{ and } 1-\mu_i = 1. \end{aligned} \tag{39}$$

All the inequalities in (38) have the form (31), and can be dealt with accordingly.

The general form of the constraints in this category is

$$\sum_{i=1}^{r} \mu_i + \sum_{j=1}^{g} q_j \geq k. \tag{40}$$

Following the reasoning for the case of (35), the above constraint must first be written as a logical expression with the same meaning. Then each of the parts of the logical expression must be manipulated as shown above. Care should be taken to simplify the set of well-formed Boolean formulas in order to avoid redundancy.

3.3.3. *Constraints containing firing vector elements only.* The last case in this section is the case when the constraint contains elements of the firing vector only. The general form is

$$\sum_{j=1}^{g} q_j \geq k. \tag{41}$$

This implies that at every firing instant at least k of the g transitions $t_1, t_2, \ldots, t_g$ must fire. These constraints must be first replaced by logical expressions having the same meaning, as in Section 3.3.2, before any further transformation is possible. In this case also, the transformations are valid for safe nets only.

As an example, consider the simpler case

$$q_1 + q_2 \geq 1. \tag{42}$$

This means that at any given firing instant one of the two transitions t_1 and t_2 must fire. Logically, this can be expressed as

$$\neg q_1 \rightarrow q_2. \tag{43}$$

Each of the qs in (43) can be substituted by the conjunction of the markings of its input places. The above then becomes

$$\neg[\mu_{1_1} \wedge \mu_{1_2} \wedge \ldots \wedge \mu_{c_1}] \rightarrow \mu_{2_1} \wedge \mu_{2_2} \wedge \ldots \wedge \mu_{c_2}, \tag{44}$$

where $\mu_{1_1}, \mu_{1_2}, \ldots, \mu_{c_1}$ and $\mu_{2_1}, \mu_{2_2}, \ldots, \mu_{c_2}$ are the sets of input places of the transitions t_1 and t_2 respectively. The expression in (44) can be written as

$$\begin{aligned} &(\neg\mu_{1_1} \rightarrow \mu_{2_1} \wedge \mu_{2_2} \wedge \ldots \wedge \mu_{c_2}) \wedge \\ &(\neg\mu_{1_2} \rightarrow \mu_{2_1} \wedge \mu_{2_2} \wedge \ldots \wedge \mu_{c_2}) \wedge \\ &\vdots \\ &(\neg\mu_{c_1} \rightarrow \mu_{2_1} \wedge \mu_{2_2} \wedge \ldots \wedge \mu_{c_2}). \end{aligned} \tag{45}$$

Each of the above has the form (15), and each can be substituted by a set of inequalities, as shown in Section 3.1.

3.4. *Equality constraints*

The last general case that remains to be considered is the case where the constraints are written as plain equalities. Different subcategories are discussed.

3.4.1. *Equality constraints containing marking vector elements only.* These constrints are written as

$$\sum_{i=1}^{r} \mu_i = k. \qquad (46)$$

This equation means that places $p_1, p_2, \ldots, p_r$ must always form a place invariant. This is really a specification for the system, and should have been incorporated into the Petri net model. If this invariant is not already part of the Petri net model, it should become part of it at this point. This is done by modifying the incidence matrix D_p of the Petri net so that (4) holds, where X contains the place invariant defined by (46). The new elements of D_p represent the arcs that should be added to the Petri net so that the place invariant becomes part of it.

3.4.2. *Equality constraints containing both marking and firing vector elements.* Assume that the constraint is of the type

$$\sum_{i=1}^{k} \mu_i + \sum_{j=1}^{g} q_j = k + g. \qquad (47)$$

This constraint means that at all times all of $\mu_1, \mu_2, \ldots, \mu_k$ should be marked and all of $q_1, q_2, \ldots, q_g$ should fire. It is rare that such a requirement is imposed on a system, but it may occur. The constraint can be transformed by substituting each of q_j with the sum of its input places and by modifying the right part accordingly. The transformation is valid for safe Petri nets only. The transformed constraint is

$$\sum_{i=1}^{k} \mu_i + \sum_{j=1}^{g} \sum_{s=1}^{c_j} \mu_{j_s} = k + \sum_{j=1}^{g} c_j, \qquad (48)$$

where c_j is the number of input places of transition t_j. The constraint now has the form (46).

3.4.3. *Equality constraints containing firing vector elements only.* Some equality constraints may contain only elements of the firing vector. A simple example is

$$q_i + q_j = 1. \qquad (49)$$

This means that one of the two transitions should fire at every instant. In other words, if q_i will not fire next then q_j must fire, and vice versa. This can be written as a logical formula

$$(q_i \rightarrow \neg q_j) \wedge (\neg q_i \rightarrow q_j). \qquad (50)$$

The same concept can be expressed by means of the input places of the two transitions:

$$\begin{gathered}[(\mu_{i_1} \wedge \mu_{i_2} \wedge \ldots \wedge \mu_{c_i}] \rightarrow \neg[\mu_{j_1} \wedge \mu_{j_2} \wedge \ldots \wedge \mu_{c_j}]) \wedge, \\ (\neg[\mu_{i_1} \wedge \mu_{i_2} \wedge \ldots \wedge \mu_{c_i}] \rightarrow \mu_{j_1} \wedge \mu_{j_2} \wedge \ldots \wedge \mu_{c_j}).\end{gathered} \qquad (51)$$

These formulas should be separated, and each should be brought into the form of (15) and then replaced by inequalities. All constraints of this type should be analyzed as shown above. This transformation is valid for safe Petri nets only.

3.5. *Graph transformations of the Petri net*

Another way of transforming constraints that contain elements of the firing vector based on a transformation performed on the Petri net graph itself is now described. Assume that a system modeled by a Petri net must satisfy the constraint

$$\mu_i + q_j \leq 1. \qquad (52)$$

This means that transition t_j cannot fire if place p_i is marked, and vice versa. In order to bring this constraint into a form that contains elements of the marking vector only, a graph transformation is performed on the process Petri net. The transition t_j is replaced by two transitions t_j and t_j' and a place p_j' beteeen them, as shown in Fig. 6.

The incidence matrix D_p of the process Petri net is increased by one column and one row, since the overall number of transitions and places of the Petri net has each been increased by one. This transformation is artificial, and does not add to or subtract anything from the Petri net model of the process. Its sole purpose is to introduce the place p_j' that records the firing of the transition t_j.

The marking μ_j' of the place p_j' replaces q_j in the constraint (52), which becomes

$$\mu_i + \mu_j' \leq 1. \qquad (53)$$

The constraint now contains only μs, and the controller can be computed as described in Section 2. Since the method produces a controller consisting of places and arcs only, no part of the controller is connected directly to the place p_j' of the transformation. After the

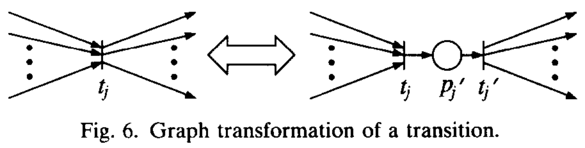

Fig. 6. Graph transformation of a transition.

controller structure has been computed by the method, the two transitions and the place of the transformation collapse to the original transition. The same transformation is performed on each and every transition that appears in the constraints. Constraints that contain only elements of the marking vector are treated in the same way.

Note that the graphical transformations do not unfold those constraints that contain elements of the firing vector, as the algebraic transformations described in Section 3 do. So the original number of constraints is conserved. However, this way of transforming constraints produces a controller that is connected directly to the corresponding transitions and does not allow these transitions to fire even if they are enabled, when their firing violates the constraint. This requires that the transitions be controllable. If they are not then the computation of the controller based on the graph transformation is not valid, and the constraints must be transformed using the algebraic methods described previously.

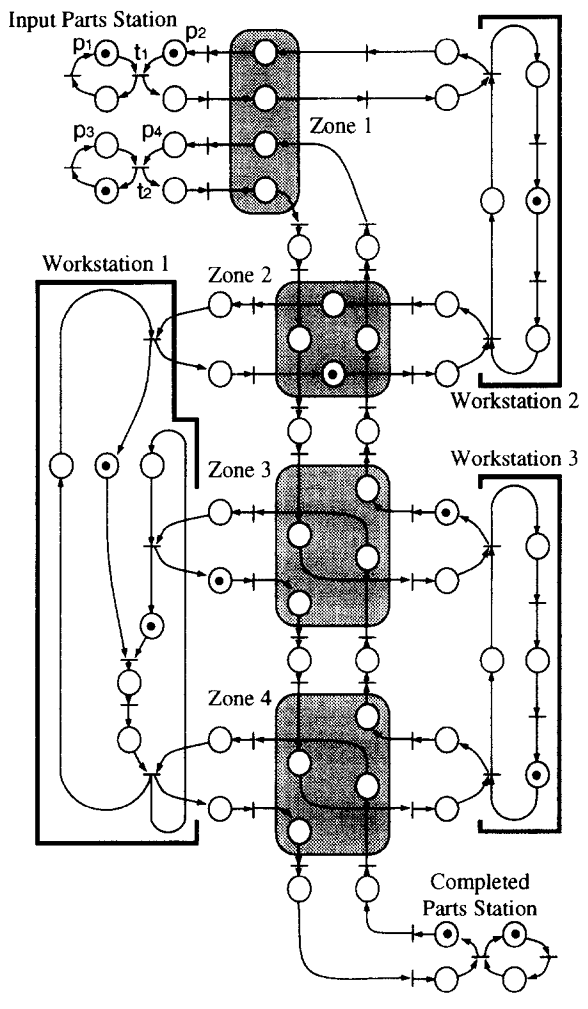

Fig. 7. The automated guided vehicle Petri net.

4. EXAMPLE

The example used to illustrate the controller computations introduced in this paper concerns a flexible manufacturing cell, used by Holloway and Krogh (1990) and Li and Wonham (1994). It consists of three workstations: two part-receiving stations and one completed-parts station. There are five automated guided vehicles (AGVs) that can transport material between the stations. The Petri net model of this system, taken from Holloway and Krogh (1990), is shown in Fig. 7. The vehicles and the parts are modeled by tokens, and the marking of the Petri net corresponds to the actual state of the system. The shaded areas represent the zones in which the vehicles' trajectories cross on the floor of the plant. A forbidden situation arises when two vehicles are present in a zone at the same time. In order to demonstrate the constraint transformation techniques described in Section 3, we introduce an additional specification for this system that is not in Holloway and Krogh (1990).

The constraints that concern the presence of the vehicles in the dangerous zones are expressed by the inequalities

$$\sum_{i\in Z_1} \mu_i \leq 1, \quad \sum_{i\in Z_2} \mu_i \leq 1, \qquad (54)$$
$$\sum_{i\in Z_3} \mu_i \leq 1, \quad \sum_{i\in Z_4} \mu_i \leq 1,$$

where Z_j is the set of indices of places that make up zone j. These constraints contain only marking vector elements, so slack variables are introduced and the inequalities become equalities:

$$\sum_{i\in Z_1} \mu_i + \mu_{c_1} = 1, \quad \sum_{i\in Z_2} \mu_i + \mu_{c_2} = 1, \qquad (55)$$
$$\sum_{i\in Z_3} \mu_i + \mu_{c_3} = 1, \quad \sum_{i\in Z_4} \mu_i + \mu_{c_4} = 1.$$

The four slack variables define four places for the controller. Each place controls the access of a zone.

The additional specification concerning the input parts stations can be written as

$$q_1 + q_2 \leq 1, \qquad (56)$$

where t_1 and t_2 are the transitions indicating that a part has been removed by an AGV from input parts stations 1 and 2 respectively. The constraint of (56) contains firing vector elements only has the form of the expression in (27), and must be transformed as described in Section 3.2.3. Both q_1 and q_2 are replaced by the sum of their input places, indicated as μ_1, μ_2, μ_3 and μ_4 in Fig. 7:

$$\mu_1 + \mu_2 + \mu_3 + \mu_4 \leq 3. \qquad (57)$$

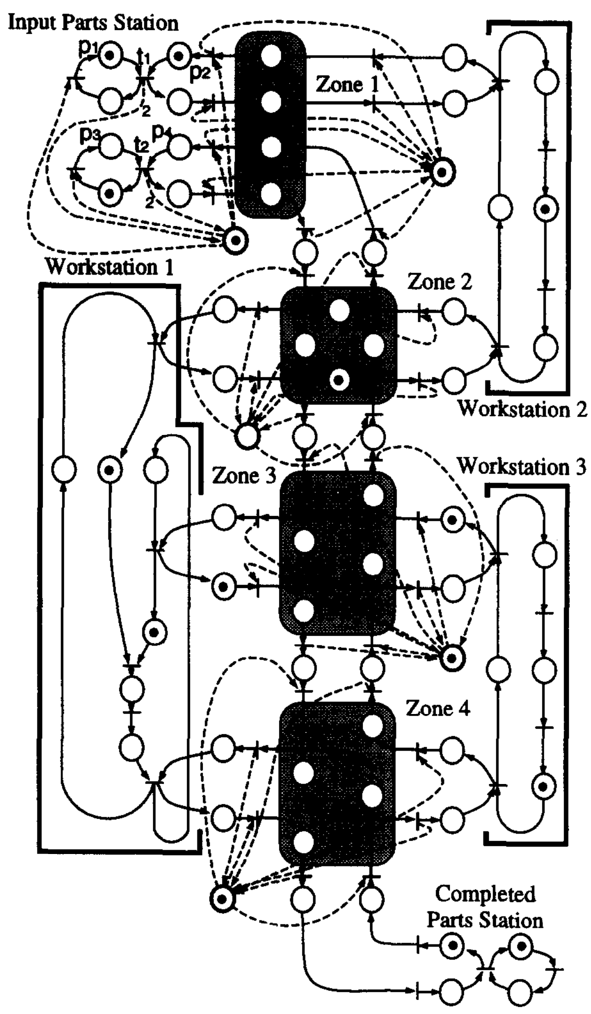

Fig. 8. The controlled AGV Petri net.

The above will not allow all four places p_1, p_2, p_3 and p_4 to be marked at the same time. An additional slack variable makes the transformed constraint an equality, and introduces a fifth controller place:

$$\mu_1 + \mu_2 + \mu_3 + \mu_4 + \mu_{c_5} = 3. \qquad (58)$$

The incidence matrix of the process is increased by five rows which correspond to the five controller places and constitute the incidence matrix D_c of the controller net. After computing D_c and μ_{c_0} from equations (7) and (8), the appropriate arcs are added to connect the controller places to the appropriate transitions of the Petri net of the process. The controlled Petri net is shown in Fig. 8.

5. CONCLUSIONS

This paper has presented a particularly simple method for constructing feedback controllers for untimed Petri nets given a set of constraints. The method is based on the idea that specifications representing desired plant behaviors can be enforced by making them invariants of the controlled net. In this paper a technique has been derived which uses place invariants representative of logical design specifications. The resulting controller consists only of places and arcs, and its size is proportional to the number of constraints. The method can accommodate a variety of constraints containing marking and/or firing vector elements. The use of place invariants makes the approach transparent and facilitates the extension of these results to more general control problems. Note that this method produces controllers identical to the monitors of Giua *et al.* (1992), which were independently derived without use of the notion of place invariants.

The method uses two different ways of transforming constraints that contain firing vector elements to the desired form. The first, based on algebraic manipulations of the constraints, can be applied to both controllable and uncontrollable transitions. In certain cases, discussed above in detail, it unfolds one constraint, replacing it by a set of constraints that have the appropriate form, thus increasing the controller size; in certain other cases this transformation technique is applicable to safe Petri nets only. The second approach is based on a graphical transformation, and it conserves the initial number of constraints. It is able to compute double-pointed arcs, and is valid for nonsafe nets as well. Its drawback is that it cannot be applied to uncontrollable transitions.

The significance of this particular approach to Petri net controller design is that a feedback controller can be computed very efficiently by a single matrix multiplication. The resulting controlled system will generally not be optimal in terms of minimizing the number of its places in the controller net. However, owing to the ease of computation, it can represent a good initial point in subsequent controller optimization. Consequently, the proposed approach appears to offer significant promise in designing Petri net feedback controllers for industrial systems.

Another advantage of the method is that it can be used to design Petri net controllers in a modular way. Assuming that the specifications on the controlled system's behavior can be decomposed into a collection of place invariants, it may be possible to realize the specifications by switching the system's marking vector between the various place invariants. The methodology presented in this paper represents a numerically efficient manner of finding controllers which enforce place invariants. This technique is therefore useful in the modular design of Petri net controllers.

There are three important characteristics of this approach. First, this method is able to construct a feedback controller without any state enumeration. This is very important, since it is now possible to design feedback controllers for very large systems. Second, the method designs a feedback controller modeled by a Petri net that is attached to the Petri net model of the process and closes the loop. The closed-loop system satisfies the control specifications. This is in contrast to other methods, which compute a control logic instead, which regulates the transition firings. Third, the method can compute controllers for general simple, untimed Petri net models, and is not restricted to cyclic or bounded Petri nets only. This is true, although some of the constraint transformations produce maximally permissive controllers only for safe Petri nets, as is indicated in Section 3.

There are several areas in which the control method based on the place invariants is oepn for further research. First the method should be expanded to deal with timed Petri nets, since these networks have added modeling power. Another important future research goal is a systematic method for transforming a set of constraints into an equivalent set when transitions in the process net are uncontrollable or unobservable.

Acknowledgements—This research was supported in part by NSF-92-16559 and IRI-91-09298.

REFERENCES

Boissel, O. (1993). Optimal feedback control design for discrete-event process systems using simulated annealing. MS thesis. University of Notre Dame.

Boucher, T. O. and M. A. Jafari (1992). Design of a factory floor sequence controller from a high level system specification. *J. Manuf. Syst.*, **11,** 401–417.

Giua, A. and F. DiCesare (1994). Blocking and controllability of Petri nets in supervisory control. *IEEE Trans. Autom. Control,* **AC-39,** 818–823.

Giua, A., F. DiCesare and M. Silva (1992). Generalized mutual exclusion constraints on nets with uncontrollable transitions. In *Proc.* 1992 *IEEE International Conference on Systems, Man, and Cybernetics*. Chicago. IL, pp. 974–979.

Holloway, L. E. and B. H. Krogh (1990). Synthesis of feedback logic for a class of controlled Petri nets. *IEEE Trans. Autom. Control,* **AC-35,** 5, 514–523.

Holloway, L. E. and B. H. Krogh (1994). Controlled Petri nets: a tutorial survey. In G. Cohen and J.-P. Quadrat (Eds), *Proc. 11th International Conf. on Analysis and Control, Discrete Event Systems*, pp. 158–168. Lecture Notes in Computer Science, Vol. 199, Springer-Verlag, Berlin.

Lautenbach, O. (1987). Linear algebraic techniques for place/transition nets. In *Advances in Petri Nets. Part I—Petri Nets: Central Models and Their Properties*, pp. 142–167. Lecture Notes in Computer Science, Vol. 254, Springer-Verlag, Berlin.

Li, Y. and W. M. Wonham (1993). Control of vector discrete-event systems I—the base model. *IEEE Trans. Autom. Control,* **AC-38,** 1214–1227 (Corrigendum (1994) **AC-39,** 1771).

Li, Y. and W. M. Wonham (1994). Control of vector discrete-event systems II—controller synthesis. *IEEE Trans. Autom. Control,* **AC-39,** 512–530.

Murata, T. (1989). Petri nets: properties, analysis, and applications. *Proc. IEEE,* **77,** 541–580.

Murata, T., N. Komoda, K. Matsumoto and K. Haruna (1986). A Petri net-based controller for flexible and maintainable sequence control and its applications in factory automation. *IEEE Trans. Ind. Electron.*, **IE-33,** 1–8.

Peterson, J. L. (1981). *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, Englewood Cliffs, NJ.

Ramadge, P. J. G. and W. M. Wonham (1989). The control of discrete event systems. *Proc. IEEE,* **77,** 81–97.

Reisig, W. (1985). *Petri Nets*, Springer-Verlag, Berlin.

Sifakis, J. (1977). Use of Petri nets for performance evaluation. In H. Beilner and E. Gelenbe (Eds), *Measuring, Modelling, and Evaluating Computer Systems.* North-Holland, Amsterdam.

Valette, R. (1986). Nets in production systems. In *Advances in Petri Nets Part II—Petri Nets: Applications and Relations to Other Models of Concurrency*, pp. 191–217. Lecture Notes in Computer Science, Vol. 255, Springer-Verlag, Berlin.

Valette, R., M. Courvoisier, H. Demmou, J. M. Bigou and C. Desclaux (1985). Putting Petri nets to work for controlling flexible manufacturing systems. In *Proc. ISCAS '85*, Kyoto, pp. 929–932.

Wonham, W. M. and P. J. Ramadge (1987). On the supremal controllable sublanguage of a given language. *SIAM J. Control Optim.*, **25,** 637–659.

Yamalidou, E. and J. C. Kantor (1991). Modelling and optimal control of discrete-event chemical processing using Petri nets. *Comput. Chem. Engng,* **15,** 503–519.

Yamalidou, E., J. O. Moody, M. D. Lemmon and P. J. Antsaklis (1994). Feedback control of Petri nets based on place invariants. Technical Report ISIS-94-002.2, ISIS Group, University of Notre Dame.

Zhou, M. C. and F. DiCesare (1989). Adaptive design of Petri net controllers for error recovery in automated manufacturing systems. *IEEE Trans. Syst. Man, Cybernetics*, **19,** 963–973.