Petri Net Supervisors for DES with Uncontrollable and Unobservable Transitions

John O. Moody, *Member, IEEE,* and Panos J. Antsaklis, *Fellow, IEEE*

***Abstract*—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 controller is described by an auxiliary Petri net connected to the plant's transitions, providing a unified Petri net model of the closed-loop system. The synthesis technique is based on the concept of admissible constraints. An inadmissible constraint cannot be directly enforced on a plant because of 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 admissible ones. When multiple transformations of this kind occur, a technique is described for creating a modified Petri net controller that enforces the union of all of these control laws. The method is practical and computationally inexpensive in terms of size, design time, and implementation complexity.**

***Index Terms*—Discrete event systems, flexible manufacturing systems, high-level synthesis, manufacturing automation, Petri nets, supervisory control.**

I. Introduction

A. Modeling DES with Petri Nets

IT IS often necessary to regulate or supervise the behavior of discrete event systems (DES) to meet safety or performance criteria, e.g., preventing automated-guided vehicles from colliding on a factory floor by restricting their access to certain mutually traveled zones. DES supervisors are used to ensure that the behavior of the plant does not violate a set of constraints under a variety of operating conditions. The regulatory actions of the supervisor are based on observations of the plant, resulting in feedback control.

It is common to see discrete event systems modeled as finite automata [22], [28]. Methods exist for designing controllers based on automata system models; however, these methods often involve exhaustive searches or simulations of system behavior, making them impractical for systems with large numbers of states and transition-causing events.

Modeling DES with Petri nets may help address some of these difficulties. Petri nets [3], [20], [21], [24] have a simple mathematical representation employing linear matrix algebra making them particularly useful for analysis and design. Petri net models are normally more compact than automata-based models that represent the same system behavior and are better suited for the representation of systems with repeated structures and flows, but large reachable state spaces. Petri nets allow for the simultaneous occurrence of multiple events without suffering from increased model complexity, as is the case with automata. In addition, they have an appealing graphical representation that makes it possible to visualize the state-flow of a system and to quickly see dependencies of one part of a system on another.

The intuitive graphical representation and the powerful algebraic formulation of Petri nets has lead to their use in a number of practical fields. Petri nets are used to model multiprocessor computer systems, computer networks, digital communication protocols, process control plants, queuing systems, and flexible manufacturing cells, among others. Often times, the graphical representation of a plant as a Petri net model is enough for an engineer to design a controller or supervisor for the plant. Many control techniques exist that involve recognizing and then manipulating certain structures that commonly appear in Petri net models. Other techniques exist for automatically verifying the reliability of these control designs. A survey of a variety of supervisory control procedures for Petri nets can be found in [8]. Representing the controller itself as a Petri net makes the verification of the combined plant/controller system simpler, as well as reducing the number of computational tools required to model the overall system. Unfortunately, even when the controller is modeled as a Petri net, this cyclic technique of design and verification can become cumbersome when the plant model is large. This process leads to the desire for an efficient method for the automatic generation of controllers based on the plant and constraint data.

B. Invariant-Based Controllers and Linear State Constraints

A method for automatically deriving supervisory controllers for DES described by Petri nets appears in [19] and [31]. The control designer is presented with a Petri net model of a DES and a set of linear constraints on the state space of the DES, and the control goal is to ensure that the constraints are met during the plant's operation. The method is based on the idea that specifications representing desired plant behaviors can be enforced by making them place invariants of the feedback controlled system. The resulting controllers are themselves Petri nets and identical

Manuscript received March 20, 1998; revised March 2, 1999. Recommended by Associate Editor, W. Lin. This work was supported in part by the National Science Foundation under Grant ECS95-31485 and by the Army Research Office under Grant DAAG55-98-1-0199.

J. O. Moody is with Lockheed Martin Federal Systems, Owego, NY 13827-3998 USA (e-mail: john.moody@lmco.com).

P. J. Antsaklis is with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556 USA (e-mail: panos.j.antsaklis.1@nd.edu).

Publisher Item Identifier S 0018-9286(00)02144-9.

to the monitors [7] of Giua *et al.*, which were derived independently using a different methodology. This technique forms the basics of the synthesis procedure described in this paper and is summarized in Section III.

Linear inequalities can be used to describe a large class of forbidden state problems. The basic synthesis procedure requires that the set of allowed states be a convex region described by the conjunction of several linear inequalities. The modified control structure of Section VI expands the class of realizable forbidden state problems by allowing nonconvex feasible regions in the form of disjunctions of linear state inequalities. Thus, the control procedure can be applied to many common problems seen in flexible manufacturing, process control, and communication networks, including serial, parallel and general mutual exclusion problems [4], [7], and the modeling and allocation of shared resources [17].

Ensuring system liveness or avoiding deadlock is a common and important supervisory control goal. The existence of liveness-ensuring supervisors for Petri nets with uncontrollable transitions has been studied by Sreenivas [26], [27]. Techniques for deadlock avoidance have been proposed by a variety of researchers; see [1], [2], [5], [10], [11], and [29] for details. These techniques involve analysis of the *siphons* or other similar structures within the Petri net plant. Often, the resulting controllers can be expressed by supervisors enforcing sets of linear inequalities on the reachable plant states. Combining these techniques with the supervision approach of this paper can then be used to prevent deadlock or ensure liveness for plants with uncontrollable and unobservable transitions.

A major goal in the field of DES control is the synthesis of supervisors under conditions where certain state-to-state transitions cannot be prevented by any action from the supervisor, i.e., conditions under which certain transitions are *uncontrollable*. The problem is then to design a controller that prevents states from occurring that violate the behavioral constraints directly or that might lead to a violation of the constraints through the action of uncontrollable transitions.

Li and Wonham [13] have made important contributions regarding the enforcement of linear constraints on Petri net plants with uncontrollable transitions. These authors show that optimal, or maximally permissive, control actions that account for uncontrollable transitions can be found by repeated applications of an integer linear programming (ILP) problem, assuming that valid control actions actually exist and that the uncontrollable portion of the net contains no loops. They also give sufficient conditions under which the solution to the ILP has a closed-form expression. In these cases, the control law can be enforced by a feedback Petri net supervisor of the type described in Section III or VI of this paper. The computation of the control law, described in Sections IV and V presented here involves only matrix algebra and is more desirable, computationally, than *analytically solving* an ILP. The tree structure assumed by Li and Wonham is only sufficient, not necessary. For example, the structure of the uncontrollable part of the plant in Section VI-B does not have a "tree structure;" in fact, it contains a loop; however, a maximally permissive supervisor was found and implemented using a modified Petri net of the kind described in Section VI.

The concept of uncontrollability is associated with the dual concept of unobservability. It is possible that a DES plant might contain certain state-to-state transitions that cannot be detected by the supervisor. The mathematical representation of these unobservable events is analogous to uncontrollable transitions. Both uncontrollable and unobservable transitions are covered by the design procedures of this paper.

C. *Summary of Contents*

Following the review of the algebraic model of Petri nets in Section II, a brief summary of the basic synthesis procedure of [31] appears in Section III. The primary contribution of this paper is the extension of these results to the synthesis of Petri net supervisors for plants with uncontrollable and unobservable transitions. One possible approach to this problem is to construct a supervisor that searches through the uncontrollably reachable markings of the plant at every iteration of the plant's evolution. This potentially expensive search is avoided here through the concept of *admissible constraints*, introduced in Section IV. A constraint is called admissible when, among the states that satisfy the constraint, none could lead (uncontrollably) to a state that does violate the constraint. Admissible constraints may be simply and directly enforced on a plant without requiring that the supervisor search through uncontrollably reachable markings. Computational techniques for generating admissible constraint transformations are presented in the Appendix, and supervisors for enforcing admissible constraints can be synthesized using the technique of Section III.

Section V shows how to characterize all admissible linear constraints for a given Petri net. When a constraint is found to be inadmissible, this characterization can be used to find the set of all admissible constraints that have feasible regions that lie within the feasible region of the original constraint. Section VI shows how to construct a supervisor that will enforce the logical union of all these admissible constraints (a disjunction of linear inequalities), thus providing for a high degree of plant freedom while accounting for uncontrollable and unobservable transitions.

Section VII shows how real-time constraints can be enforced by extending the supervision method to timed Petri nets. Concluding remarks are given in Section VIII.

II. Petri Net Fundamentals

A Petri net is a directed bipartite graph. The structure of a Petri net is described by (P, T, D^+, D^-), where P and T are disjoint sets representing the vertices of the graph, known as *places* and *transitions*, and D^+ and D^- are integer matrices with nonnegative elements representing the flow relation between the two vertex types.

Places in a Petri net hold *tokens*, the distribution of which indicates the net's state or its *marking*. Transitions direct the flow of tokens between places, thus the *firing* of a transition is a state-changing *event* in a DES model. A Petri net's *incidence matrix* represents the weighted connections of directed arcs between its places and transitions. It is composed of two matrices,

D^-, representing arcs from places to transitions, and D^+, representing arcs from transitions to places

$$D = D^+ - D^- \qquad (D^+, D^- \geq 0).$$

The incidence matrix is used to construct a difference equation that describes the evolution of the net's state

$$\begin{aligned} \mu(k+1) &= \mu(k) + Dq(k) \\ \mu(0) &= \mu_0 \end{aligned} \tag{1}$$

where

$$\mu \in \mathbb{Z}^n, \ q \in \mathbb{Z}^m, \ \mu, q \geq 0 \tag{2}$$

where

$\mathbb{Z}$	set of integers;
μ	net's state or *marking vector*;
q	input or *firing vector*;

and $D \in \mathbb{Z}^{n \times m}$. The notation $\mu, q \geq 0$ indicates that every element of the marking and firing vectors is nonnegative at all times. This element-by-element interpretation of inequalities will be used throughout this paper whenever vectors or matrices appear on either side of the inequality symbol.

The nonnegativity conditions in (2) lead to the Petri net *transition enabling rule.* A firing vector q is feasible (represents a valid set of transition firings) if $q \geq 0$ and

$$D^- q \leq \mu \tag{3}$$

where the iteration counter k has been dropped for convenience. If the Petri net's transitions contain no self loops, i.e., the positions of the nonzero elements in D^+ and D^- are mutually exclusive, then the transition enabling rule can be written as

$$\mu + Dq \geq 0. \tag{4}$$

Care must be taken when using (3) or (4) when q indicates the concurrent firing of multiple transitions. A variety of different techniques exist for handling concurrency. Concurrency may not be allowed at all, in which case q would be a zero vector with a single element equal to one. Concurrency may be allowed only when each of the indicated transition firings could occur one after the other in any order. In this case, q must satisfy (3), or each transition firing indicated in q must independently satisfy (4) as well as the complete q vector. If (4) is used without this check for independently enabled transitions, then certain concurrent firings may be allowed even though some or all of the individual transitions indicated in the firing could not fire by themselves. The choice of which of these methods to use is dictated by the modeling requirements and the particular plant.

Petri net *place invariants* are fundamental to the supervisor synthesis technique described in the following section. A place invariant is an integer vector x that satisfies

$$\forall \text{ reachable } \mu, \ x^T \mu = x^T \mu_0.$$

Thus, $x^T \mu$ is a constant for all reachable states if x is a place invariant. Place invariants can be computed by finding solutions to the equation

$$x^T D = 0.$$

III. Invariant-Based Control Synthesis

A. Description of Method

The system in need of supervision, the *plant net,* is modeled by a Petri net with n places and m transitions. The plant's incidence matrix is $D_p \in \mathbb{Z}^{n \times m}$. The *controller net* is a Petri net with incidence matrix $D_c \in \mathbb{Z}^{n_c \times m}$ made of the plant's transitions and a separate set of places. The *controlled system* or *controlled net* is the Petri net with incidence matrix $D \in \mathbb{Z}^{(n+n_c) \times m}$ made of both the original plant and the added controller. The control goal is to force the plant to obey constraints of the form

$$L\mu_p \leq b \tag{5}$$

where $\mu_p \in \mathbb{Z}^n$, $\mu_p \geq 0$ is the marking vector of the plant, $L \in \mathbb{Z}^{n_c \times n}$, and $b \in \mathbb{Z}^{n_c}$. 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 n_c separate linear inequalities.

Inequality (5) can be transformed into an equality by introducing an external Petri net controller that contains places that represent nonnegative "slack variables." The constraint then becomes

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

where $\mu_c \in \mathbb{Z}^{n_c}$, $\mu_c \geq 0$ is the marking of the controller. Note that $\mu_c \geq 0$ because the number of tokens in a Petri net place cannot become negative; thus, (6) implies inequality (5). The closed-loop system has the following Petri net structure:

$$D = \begin{bmatrix} D_p \\ D_c \end{bmatrix}, \qquad \mu_0 = \begin{bmatrix} \mu_{p_0} \\ \mu_{c_0} \end{bmatrix}. \tag{7}$$

The controller is computed by observing that the introduction of slack variables forces a set of place invariants on the overall controlled system defined by (6). The results in the propositions below have been introduced and discussed in [15], [18], [19], and [31].

Theorem 1—Invariant-Based Controller Synthesis: If

$$b - L\mu_{p_0} \geq 0 \tag{8}$$

then a Petri net controller, $D_c \in \mathbb{Z}^{n_c \times m}$ with initial marking $\mu_{c_0} \in \mathbb{Z}^{n_c}$

$$D_c = -LD_p \tag{9}$$

$$\mu_{c_0} = b - L\mu_{p_0} \tag{10}$$

enforces constraint (5) when included in the closed-loop system (7), assuming that the plant's transitions are controllable and observable.

If inequality (8) is not true, then the constraints cannot be enforced by any controller because the initial conditions of the plant lie outside the range allowed by the constraints.

Proof: If inequality (8) is not true, then obviously $L\mu_{p_0} \not\leq b$, and at least one row of L, l_i exists, such that $l_i^T \mu_{p_0} > b_i$ and the initial conditions of the plant violate the constraint. If the inequality is true, then (10) shows that the initial conditions of

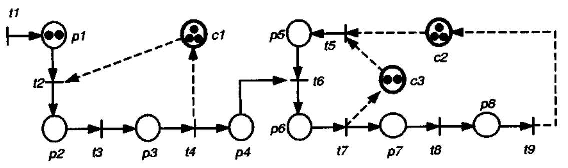

Fig. 1. The piston rod robotic assembly cell with its initial controller.

TABLE I
PLACE DESCRIPTIONS FOR FIG. 1

p_1	Engine block and crank shaft are ready to be processed.
p_2	S-380 robot aligns the crank shaft.
p_3	S-380 robot picks up new piston rod and positions it in work space.
p_4	Engine block prepared and ready for work by M-1 robot.
p_5	M-1 robot picks up a piston pulling tool.
p_6	M-1 robot pulls the piston rod into the engine block, returns pulling tool.
p_7	M-1 robot positions a cap on the piston rod.
p_8	M-1 robot bolts the cap to the piston rod.

the controller are defined as a vector representing the slack in each of the constraints represented by $L\mu_p \leq b$.

Equation (9) forces (6) to be place invariants of the closed-loop system (see [18], [19], and [31]), thus inequality (5) will be true for all reachable markings of the closed-loop system. □

Proposition 2—Invariant-Based Controllers are Maximally Permissive: Given a plant and a set of enforceable constraints (5), a controller constructed according to the rules of Theorem 1 only acts to disable transitions when the firing indicated by the given q vector leads to a state forbidden by (5).

Proof: According to the rules of Petri net evolution, the controller will only disable transitions when the firing indicated by q would cause the marking of at least one of its places to become negative. Because (6) represents a set of invariants in the closed-loop system, any negative element in μ_c indicates a violation of (5), and all states allowed by (5) correspond to nonnegative values in μ_c.

It has also been shown in [31] that the controller only induces place invariants in the closed-loop system that are specifically described by (6). All place invariants of the closed-loop system are accounted for by those originally present in the plant and those specifically required to enforce the constraints. □

B. Example—The Piston Rod Robotic Assembly Cell

The piston rod assembly cell application is partially based on a similar plant described in [4, ch. 8]. The Petri net model of the plant is shown in Fig. 1, and Table I details the meaning of each place in the net. The number of tokens in each place signifies the number of resources or robots engaged in the activities described in Table I. The assembly of each part requires work by two different robots. An S-380 robot is used to prepare and align the parts for assembly, and an M-1 robot installs the cap on the piston rod. Places c_1, c_2, and c_3 in Fig. 1 are the components of a supervisory controller, the design of which is covered here.

Three S-380 and three M-1 robots are available in the assembly cell. Two piston pulling tools exist. These resource constraints are translated into linear inequalities on the state space of the plant

$$\mu_2 + \mu_3 \leq 3 \quad \text{(three S-380 robots)} \tag{11}$$

$$\mu_5 + \mu_6 + \mu_7 + \mu_8 \leq 3 \quad \text{(three M-1 robots)} \tag{12}$$

$$\mu_5 + \mu_6 \leq 2 \quad \text{(two piston pulling tools).} \tag{13}$$

Each inequality is enforced by a separate controller place. The connections of these three places, c_1, c_2, and c_3, to the plant and their initial markings are calculated using Theorem 1, resulting in the maximally permissive supervisor shown in Fig. 1

$$D_c = -\underbrace{\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}}_{L}$$

$$\times \underbrace{\begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}}_{D_p}$$

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

$$\mu_{c0} = \underbrace{\begin{bmatrix} 3 \\ 3 \\ 2 \end{bmatrix}}_{b} - L\mu_{p0} = \begin{bmatrix} 3 \\ 3 \\ 2 \end{bmatrix}.$$

IV. ADMISSIBLE CONSTRAINTS AND CONTROLS

A. Uncontrollable and Unobservable Transitions

A transition is called *uncontrollable* if the firing of that transition may not be inhibited by an external action. The freedom of an uncontrollable transition to fire is limited solely by the structure and state of the plant.

In order for a Petri net controller to inhibit a transition, it must contain an arc from a controller place to the transition. The transition will be disabled if the number of tokens in the control place is less than the arc weight.

A transition is called *unobservable* if the firings of that transition cannot be directly detected or measured. Because the firing of an unobservable transition cannot be detected, a controller state change cannot be triggered by such a firing.

For a Petri-net–based controller, both input and output arcs to the plant transitions are used to trigger state changes in the controller. *A Petri net controller cannot have any connections to an unobservable transition, thus all unobservable transitions are also implicitly uncontrollable;* of course, an uncontrollable transition may or may not be unobservable. We can imagine a situation in which the occurrence of some event in a plant could be blocked without the controller ever receiving any feedback relating directly to that event, but, in practical situations, the ability to inhibit an event is usually coupled with the ability to detect occurrences of that event. For this reason, this limitation on Petri-net–based controllers is not too severe.

B. Constraint Transformations

Given a set of constraints, $L\mu_p \leq b$, a supervisor must work to ensure that the constraints are never violated directly and may never be violated through the firing of uncontrollable transitions or through incomplete knowledge because of unobservable transitions. In order to avoid expensive online searches by the supervisor through the uncontrollably reachable markings of the plant, the approach taken here is to actually modify the constraints themselves such that the new constraints account for uncontrollability and unobservability. The following definitions are useful in understanding the motivation for the transformation of constraints. The definitions are with respect to a plant with possible uncontrollable or unobservable transitions and constraints on the marking behavior of the plant in the form $L\mu_p \leq b$. Unobservable transitions are also assumed to be uncontrollable.

Definition 3: An *admissible marking* μ_p is a marking such that the following occurs.

1) $L\mu_p \leq b$.
2) For all markings μ_p' reachable from μ_p through the firing of uncontrollable transitions, $L\mu_p' \leq b$.

If either of these conditions is not met, then the marking is *inadmissible*.

Definition 4: Given a plant with initial marking $\mu_p(0) = \mu_{p_0}$, an *admissible constraint* satisfies two conditions.

1) $L\mu_{p_0} \leq b$.
2) For all $\mu_p(N)$ reachable from $\mu_p(0)$ through any path of consecutively reachable markings, $\mu_p(0) \rightarrow \mu_p(1) \rightarrow \cdots \rightarrow \mu_p(N)$, where

$$L\mu_p(i) \leq b, \qquad \text{for } 1 \leq i \leq N$$

$\mu_p(N)$ is an admissible marking.

If a constraint does not satisfy both of these conditions, then it is *inadmissible*.

If a constraint is admissible, then condition 2) of Definition 4 indicates that the firing of uncontrollable transitions can never lead from a state that satisfies the constraint to a new state that violates that constraint. Note that the admissibility of a particular marking does not imply that the marking is actually reachable, either because of the initial marking of the plant or because of the restrictions of a supervisor.

An admissible constraint will only allow admissible markings; however, admissible markings may exist that could be reached by the uncontrolled plant that cannot be reached under maximally permissive supervision. Definition 4 incorporates this by checking the admissibility of markings that were achieved by following paths in which all intermediate markings satisfy the constraint. This set of reachable, admissible markings is similar to the set $\mathrm{Re}(G, P)$ defined in [12].

Example: The Petri net of Fig. 2 contains two uncontrollable transitions: t_2 and t_3. Tokens in places p_2 and p_3 cannot be prevented from freely traveling between these two places. However, t_1 can be used to stop the introduction of new tokens into p_2 and p_3, and t_4 can be used to prevent tokens from leaving.

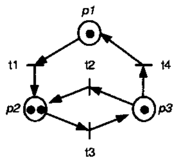

Fig. 2. Transitions 2 and 3 are uncontrollable.

The constraint

$$\mu_3 \leq 1 \tag{14}$$

is inadmissible. The initial state of the plant $\mu_0 = [1 \quad 2 \quad 1]^T$ satisfies the constraint, but the uncontrollable firing of t_3 would lead to the state $\mu = [1 \quad 1 \quad 2]^T$, which violates (14). The constraint fails condition 2) of Definition 4.

The constraint

$$\mu_1 \leq 1 \tag{15}$$

is admissible. The current state of the plant satisfies the constraint, and for any state that satisfies the constraint, no firing of uncontrollable transitions occurs that would lead to a state that does not satisfy it. The marking of p_1 is affected only by the firings of transitions t_1 and t_4, both of which are controllable.

If $L\mu_p \leq b$ is inadmissible, then it is desirable to find another constraint $L'\mu_p \leq b'$ such that $L'\mu_p \leq b'$ is an admissible constraint, and for all μ_p such that $L'\mu_p \leq b'$, $L\mu_p \leq b$ is also true. In the example above, we could replace constraint (14) with

$$\mu_2 + \mu_3 \leq 1. \tag{16}$$

This constraint is admissible according to Definition 4, and all reachable states that satisfy (16) also satisfy (14). Thus, constraint (14) could be enforced by designing a controller for constraint (16) using the technique of Section III. Unfortunately, a controller designed this way may not be maximally permissive. The method of handling uncontrollable/unobservable transitions in Section V of this paper follows along these lines, but it also includes the idea of finding *all constraints* $L'\mu_p \leq b'$ that meet the criteria above. Section VI then shows how to construct a controller that enforces the disjunction of these inequalities, allowing for a high degree of plant freedom.

C. Petri-Net–Modeled Supervisors

The supervisors used in this paper are modeled by Petri nets. Uncontrollable and unobservable transitions can cause problems for Petri-net–based supervisors because of limitations in their modeling power; however, Petri net supervisors are still useful for several reasons. Unified plant/controller models are elegant, facilitating implementation and closed-loop system analysis. The evolution of Petri net models is inexpensive to compute, facilitating their use in real-time control applications. Desirable Petri net qualities, such as automatic handling of concurrent events, are maintained with unified plant/controller Petri net models. Though the decision power of a Petri net supervisor is not unlimited, a good variety of DES control problems can be effectively and efficiently solved through their use. Recognizing the controller as a Petri net facilitates understanding of what can and cannot be done with the supervisor, as will become evident in the material below.

For an invariant-based Petri net supervisor to be realizable on a plant with uncontrollable and unobservable transitions, the constraint it is enforcing must be admissible. Proposition 5 provides necessary and sufficient conditions for any behavioral constraint to be admissible. Definitions 3 and 4 are written specifically for linear state-based constraints; however, they can be thought of in terms of general behavioral constraints. That is, Definition 3 requires that a particular marking and all behaviors achieved through the firing of uncontrollable transitions from that marking conform to the constraint. Definition 4 requires that the initial condition of a plant satisfy the constraint and that all markings visited through any behavior that conforms to the constraint are admissible markings.

The behavior of a maximally permissive supervisor is analyzed in Proposition 5. Here, the term maximally permissive is used in the sense of Section III, where all transitions are assumed to be controllable. In this case, a maximally permissive controller only prevents firings that lead to states that directly violate the given constraint.

Proposition 5—General Constraint Admissibility: A constraint on the marking or firing behavior of a Petri net is *admissible* iff the following occurs.

1) Initial conditions of the plant satisfy the constraint.
2) A maximally permissive controller (constructed under the assumption that all transitions are controllable) exists that enforces the constraint and does not inhibit any uncontrollable transitions that would otherwise be enabled.

Proof: Clearly, if the initial conditions of a plant violate a constraint, then that constraint cannot be enforced and is inadmissible according to condition 1) of Definition 4. Furthermore, if the constraint is admissible, then a maximally permissive controller would have no need to attempt to disable otherwise enabled uncontrollable transitions, as per Definition 4.

A maximally permissive controller will only allow reachable states or behaviors that do not violate the constraint. Thus, if a maximally permissive controller never attempts to inhibit an otherwise enabled uncontrollable transition, then the constraint it is enforcing is admissible according to Definition 4. □

Corollary 6—Place-Constraint Admissibility: The single vector constraint $l^T\mu_p \leq b$ is admissible iff the controller with incidence matrix $D_c = -l^T D_p$ and initial marking $\mu_{c0} = b - l^T\mu_{p0} \geq 0$ will never attempt to disable an uncontrollable transition that would otherwise be enabled.

Proof: If $\mu_{c0} \ngeq 0$, then the initial conditions of the plant violate the constraint, and that constraint cannot be enforced and is inadmissible according to condition 1 of Definition 4. Invariant-based controllers are maximally permissive according to Proposition 2; if the constraint is admissible, then this maximally permissive controller would have no need to attempt to disable otherwise enabled uncontrollable transitions, as per Definition 4.

Invariant-based controllers only allow reachable states that do not violate the constraint by inhibiting the firing of any transition that would directly lead to a marking that violates the constraint. Thus, if it never attempts to inhibit an otherwise enabled uncontrollable transition, then the constraint it is enforcing is admissible according to Definition 4. □

Remark: Corollary 6 deals with individual inequality constraints instead of the vector inequality $L\mu_p \leq b$ because each of the inequalities in $L\mu_p \leq b$ can be handled independently. Certain constraints in $L\mu_p \leq b$ may be admissible, and others may not.

Equations (9) and (10) from Theorem 1 show that it is possible to construct the incidence matrix D_c of a maximally permissive Petri net controller as a linear combination of the rows of the incidence matrix of the plant. Negative elements in D_c correspond to arcs from controller places to plant transitions. These arcs act to inhibit plant transitions when the corresponding controller places are empty, and thus, they can only be applied to plant transitions that permit such external control. If we group all of the columns of D_p that correspond to transitions that cannot be controlled into the matrix D_{uc}, we obtain the following corollary.

Corollary 7—$l^T D_{uc} \leq 0$ Implies Admissibility: Given a plant with uncontrollable transitions described by the incidence matrix D_{uc} and a constraint $l^T\mu_p \leq b$, if

$$l^T D_{uc} \leq 0 \tag{17}$$

then the constraint is admissible for the given plant.

Proof: The proof follows from Corollary 6 and the construction of the Petri net controller whose incidence matrix is $D_c = -l^T D_p$ as described in Section IV-A. Inequality (17) ensures that the controller draws no arcs to uncontrollable transitions. □

Example: Corollary 7 can be used to verify the results from the example in Section IV-B. Because transitions t_2 and t_3 are uncontrollable in the Petri net of Fig. 2, D_{uc} is composed of the second and third columns of the plant incidence matrix

$$D_p = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & 1 & -1 & 0 \\ 0 & \underbrace{-1 & 1}_{D_{uc}} & -1 \end{bmatrix}.$$

Constraint (14) fails to meet condition (17) of the corollary

$$[0 \quad 0 \quad 1]D_{uc} = [-1 \quad 1].$$

Constraints (15) and (16) both meet condition (17) and are both admissible

$$\begin{aligned}[1 \quad 0 \quad 0]D_{uc} &= [0 \quad 0] \\ [0 \quad 1 \quad 1]D_{uc} &= [0 \quad 0].\end{aligned}$$

Remark: Corollary 7 provides only a sufficient condition for constraint admissibility. Situations exist for which $l^T D_{uc} \nleq 0$, but $l^T\mu_p \leq b$ is still an admissible constraint (see [15] and [16]). However, for most practical examples, constraints that fail condition (17) are inadmissible and will need to be transformed if they are to be enforced.

As discussed in Section IV-A, it is illegal for the controller to change its state based on the firing of an unobservable transition, because no direct way exists for the controller to be told that such a transition has fired. Both input and output arcs from the controller places are used to change the controller state based on the firings of plant transitions. Let the matrix D_{uo} represent the incidence matrix of the unobservable portion of the Petri net. This matrix is composed of the columns of D_p that correspond to unobservable transitions, just as D_{uc} is composed of the uncontrollable columns of D_p.

Corollary 8—$l^T D_{uo} = 0$ Implies Admissibility: Given a plant with unobservable transitions described by the incidence matrix D_{uo} and a constraint $l^T \mu_p \leq b$, if

$$l^T D_{uo} = 0 \tag{18}$$

then the constraint is admissible.

Proof: As with Corollary 7, the proof follows from Corollary 6 and the construction of the Petri net controller whose incidence matrix is $D_c = -l^T D_p$ as described in Section IV-A. Equation (18) ensures that the controller draws no arcs to or from unobservable transitions. □

Remark: Corollaries 7 and 8 indicate that it is possible to observe a transition that we cannot inhibit, but it is illegal to directly inhibit a transition that we cannot observe.

Suppose, given a set of constraints $L\mu_p \leq b$, we construct the matrices LD_{uc} and LD_{uo} and observe that violations to conditions (17) or (18) exist. What other constraints, of the form $L'\mu_p \leq b'$, will also maintain the original constraint $L\mu_p \leq b$?

Lemma 9—Constraint Transformation Structure:

Let $R_1 \in \mathbb{Z}^{n_c \times n}$ satisfy $R_1 \mu_p \geq 0 \quad \forall \mu_p$ (19)

Let $R_2 \in \mathbb{Z}^{n_c \times n_c}$ be a positive-definite 3 diagonal matrix. (20)

If $L'\mu_p \leq b'$, where

$$L' = R_1 + R_2 L \tag{21}$$

$$b' = R_2(b + \mathbf{1}) - \mathbf{1} \tag{22}$$

and $\mathbf{1}$ is an n_c dimensional vector of 1's, then $L\mu_p \leq b$.

Proof: The transformed constraint is $(R_1 + R_2 L)\mu_p \leq R_2(b + \mathbf{1}) - \mathbf{1}$. Because all of the elements are integers, the inequality can be transformed into a strict inequality

$$(R_1 + R_2 L)\mu_p < R_2(b + \mathbf{1}).$$

Because R_2 is diagonal and positive definite

$$R_2^{-1} R_1 \mu_p + L\mu_p < b + \mathbf{1}.$$

Assumptions (19) and (20) imply that all elements of the vector $R_2^{-1} R_1 \mu_p \geq 0$; therefore, $L\mu_p \leq b$. □

Lemma 9 shows a class of constraints, $L'\mu_p \leq b'$, which, if enforced, will imply that $L\mu_p \leq b$ is also enforced. The following lemma is used to show the conditions under which a particular set of constraints can be enforced on a particular Petri net.

Lemma 10—Initial Condition Check for Transformed Constraints: The constraint $L'\mu_p \leq b'$, where $L' \neq 0$ and b' are defined by (21) and (22), can be enforced on a Petri net with initial marking μ_{p_0} iff

$$0 \leq R_1 \mu_{p_0} \leq R_2(b + \mathbf{1} - L\mu_{p_0}) - \mathbf{1}. \tag{23}$$

Proof: Substituting L' and b' into (23) gives $0 \leq b' - L'\mu_{p_0}$, which is equivalent to the condition $L'\mu_{p_0} \leq b'$ that states that the initial conditions of the plant must fall within the acceptable region of the constraints. Clearly, if a controller does exist, then the initial conditions of the plant must not violate the constraints. Furthermore, as shown in Section III, if the initial conditions lie within the acceptable region of the plant [inequality (8)], a controller to enforce the conditions can be computed with incidence matrix $D_c = -L'D_p$ and initial marking $\mu_{c_0} = b' - L'\mu_{p_0}$. □

Theorem 11 combines Corollaries 7 and 8 with the conditions for creating a valid set of transformed constraints in Lemmas 9 and 10 to show how to construct a Petri net controller.

Theorem 11—Constraint Transformation and Supervisor Synthesis: Let a plant Petri net with incidence matrix D_p be given with a set of uncontrollable transitions described by D_{uc} and a set of unobservable transitions described by D_{uo}. A set of linear constraints on the net marking, $L\mu_p \leq b$, are to be imposed. Assume R_1 and R_2 meet (19) and (20) with $R_1 + R_2 L \neq 0$, and let

$$\begin{aligned} &[R_1 \quad R_2] \begin{bmatrix} D_{uc} & D_{uo} & -D_{uo} & \mu_{p_0} \\ LD_{uc} & LD_{uo} & -LD_{uo} & L\mu_{p_0} - b - \mathbf{1} \end{bmatrix} \\ &\quad \leq [0 \quad 0 \quad 0 \quad -\mathbf{1}]. \end{aligned} \tag{24}$$

Then the controller

$$D_c = -(R_1 + R_2 L)D_p = -L'D_p \tag{25}$$

$$\mu_{c_0} = R_2(b + \mathbf{1}) - \mathbf{1} - (R_1 + R_2 L)\mu_{p_0} = b' - L'\mu_{p_0} \tag{26}$$

exists and causes all subsequent markings of the closed-loop system (7) to satisfy the constraint $L\mu_p \leq b$ without attempting to inhibit uncontrollable transitions and without detecting unobservable transitions.

Proof: According to (9) and (10), (25) and (26) define a controller that enforces the constraint $L'\mu_p \leq b'$. Lemma 9 shows that, if assumptions (19) and (20) are met, then a controller that enforces a particular constraint $L'\mu_p \leq b'$ will also enforce the constraint $L\mu_p \leq b$. The fourth column of inequality (24) indicates that the condition in lemma 10 is satisfied; thus, the controller exists and the control law can be enforced. The first column of (24) indicates that $L'D_{uc} \leq 0$; thus, condition (17) is satisfied and no controller arcs are drawn to the uncontrollable transitions. The second and third columns of (24) indicate that $L'D_{uo} = 0$; thus, condition (18) is satisfied and no arcs are drawn between the controller places and the unobservable plant transitions. □

Remark: $[R_1 \quad R_2]$, which is used to describe the constraint transformation, is a left multiplier in (24); thus, this matrix represents the use of rows from D_{uc} to eliminate positive elements from LD_{uc} and the use of rows from D_{uo} to zero the elements of LD_{uo}.

The usefulness of Theorem 11 for specifying controllers to handle plants with uncontrollable and unobservable transitions lies in the ease in which the matrices R_1 and R_2 can be generated. Two computational techniques for computing these matrices can be found in the Appendix. The first technique is an integer program that searches through feasible solutions satisfying (17) and (18) along directions dictated by (23). The second technique finds appropriate transformations through the use of

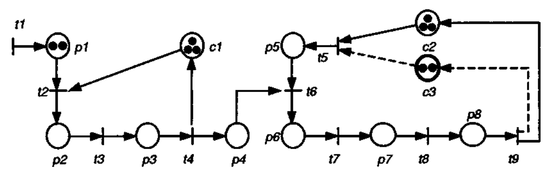

Fig. 3. The modified assembly cell and supervisor after the introductions of uncontrollable and unobservable transitions.

constrained integer row operations. Full details of each algorithm are included in the Appendix.

D. Example—Uncontrollable and Unobservable Transitions in the Assembly Cell

Uncontrollable and unobservable transitions are introduced into the robotic assembly cell example from Section III-B. The operation of the M-1 robots is now considered to be governed by a separate, independent controller. Transitions t_6, t_7, and t_8 can neither be observed nor inhibited by the resource supervisor of Section III-B.

The uncontrollable and unobservable portion of the plant is described by the matrix D_{uo}, which is composed of the sixth through eighth columns of D_p. Of the three constraints, (11)–(13), only (13) fails the test of Corollary 8, because

$$[0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0]D_{uo} = [0 \quad -1 \quad 0]. \tag{27}$$

If the plant transitions were merely uncontrollable and not unobservable as well, then the constraint would be admissible according to Corollary 7, but Corollary 8 indicates that $l^T D_{uo} = 0$ is the sufficient condition for admissibility with unobservable transitions.

The right-hand side of (27) can be zeroed by adding to it the seventh and eighth rows of D_{uc}. In terms of Theorem 11, this process corresponds to a constraint transformation using

$$R_1 = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1]$$
$$R_2 = 1.$$

The transformed constraint

$$\mu_5 + \mu_6 + \mu_7 + \mu_8 \leq 2 \tag{28}$$

is admissible and represents the maximally permissive admissible control law for enforcing (12). The new configuration for c_3 is shown in Fig. 3. Note that places c_2 and c_3 now have the same connections to the plant. It would be possible to eliminate c_2, because its action is now implied by c_3, but instead, both places will be maintained to account for dynamic changes in the number of available resources or sensors (see [16] and [17]).

V. Structure of Admissible Constraints and Controls

Given a plant with uncontrollable/unobservable transitions, it is useful to seek methods for transforming inadmissible constraints into admissible ones, but it is also logical to ask, in general, what are the admissible constraints for this plant? Does a way exist to characterize all or most of these constraints? Section V-A provides a method for just such a characterization. Section V-B shows how this characterization may be used to synthesize controllers for plants with unobservable transitions, and Section V-C covers the synthesis problem for plants with uncontrollable transitions.

A. Characterization of Admissible Constraints

As in the previous sections, let the matrix D_{uo} represent the incidence matrix of the unobservable portion of the Petri net. It is illegal for the controller $D_c = -LD_p$ to contain any arcs in the unobservable portion of the net; thus, an admissible set of constraints will satisfy

$$LD_{uo} = 0 \tag{29}$$

as indicated in Corollary 8.

Any L that satisfies (29) will lie within the kernel of D_{uo}. Let X satisfy

$$XD_{uo} = 0 \tag{30}$$

where $X \in \mathbb{Z}^{(n-\operatorname{rank} D_{uo})\times n}$. The rows of X form a linearly independent basis for the kernel of D_{uo} (X is full rank). The process of finding X is equivalent to finding the place invariants (an algorithm appears in [14]) of the unobservable portion of the plant Petri net. All realizable constraints must lie within the basis described by the rows of X, and thus can be formed as linear combinations of these rows. Every admissible constraint can be described by $k^T X$, where $k \in \mathbb{Z}^{(n-\operatorname{rank} D_{uo})}$. In general, the coefficient matrix of any set of admissible constraints $L' \in \mathbb{Z}^{n_c \times n}$ can be written as

$$L' = KX \tag{31}$$

where $K \in \mathbb{Z}^{n_c \times (n-\operatorname{rank} D_{uo})}$.

A characterization of all admissible constraints is not as transparent for the case of uncontrollable transitions as it is for unobservable ones. For unobservable transitions, we have an equality, $LD_{uo} = 0$, which must be satisfied, but for uncontrollable transitions, it is an inequality, $LD_{uc} \leq 0$, so we cannot simply find the kernel of D_{uc}. In this case, the following equality can be formed

$$LD_{uc} + \Delta = 0$$

where Δ is a matrix of nonnegative slack variables. The previous equation is rewritten as

$$[L \quad \Delta]\begin{bmatrix} D_{uc} \\ I \end{bmatrix} = 0.$$

A kernel X, solving

$$X\begin{bmatrix} D_{uc} \\ I \end{bmatrix} = 0$$

can then be used to construct a basis for all admissible linear constraints that may be placed on the plant. X must be computed so as to ensure that the elements that correspond to Δ are nonnegative. Each element of the kernel will have $n + n_{uc}$ scalar components, where n_{uc} is the number of uncontrollable transitions. The final n_{uc} elements of each kernel vector correspond to the slack variables in Δ. Additional row operations may need to be performed on X to ensure that the final n_{uc} elements in each vector are nonnegative. After ensuring that none of the slack variables are negative, all admissible constraint matrices L can be found in the the linear combinations of X that leave nonnegative values in the final n_{uc} slack columns. The first n components of a given kernel vector represent an admissible value for a row of L, as long as the final n_{uc} components of the kernel vector are nonnegative.

B. Constraint Transformations for Unobservability

Suppose we have a set of constraints $L\mu \leq b$ such that $LD_{uo} \neq 0$. It is necessary to create new constraint matrices (L', b') with the following two properties.

1) $L'D_{uo} = 0$.
2) $\forall \mu_p, L'\mu_p \leq b' \rightarrow L\mu_p \leq b$.

Property 1) is necessary to ensure that the new controller will not utilize the unobservable transitions, and property 2) indicates that the new constraints must be at least as restrictive as the original ones. Lemma 9 from Section IV-C is used to deal with this condition.

To perform the transformation, it is necessary to determine values for the matrices R_1 and R_2 defined in Lemma 9 that meet assumptions (19) and (20). It is possible for a designer to determine the values of R_1 and R_2 by using the kernel of D_{uo}. Combining (21) and (31), we see that

$$L' = KX = R_1 + R_2L.$$

The designer should premultiply each constraint in L by some positive integer (which will determine the diagonal elements in R_2) and add new positive coefficients (which will determine R_1) such that the new constraint is a linear combination of the rows of X. This process will yield the L' matrix, and b' can be calculated using R_2 and (22).

When multiple distinct transformations exist, the technique of Section VI can be used to enforce the disjunction of all these inequalities.

C. Constraint Transformations for Uncontrollability

The invariant-based control method yields maximally permissive supervisors for enforcing linear constraints of the form $L\mu_p \leq b$. When these constraints are transformed, because of the uncontrollability and unobservability of certain transitions, into $L'\mu_p \leq b'$, the invariant-based control method will still yield a maximally permissive realization of the transformed constraint. Unfortunately, the new constraint itself may not represent the most permissive admissible control law corresponding to the original constraint. The maximally permissive admissible constraint associated with a linear predicate on the plant's marking may be a nonlinear predicate that cannot be optimally controlled by a standard invariant-based controller.

At this time, a complete description of the conditions under which an optimal transformation of linear constraints is another set of linear constraints is unknown. Li and Wonham [13] have shown that, when the uncontrollable portion of the plant has a "type 1 tree structure," the optimal transformation will be a disjunction of linear predicates,[1] whereas a "type 2" structure will yield a linear transformation. These conditions, however, are only sufficient, not necessary.

[1]A procedure for enforcing these with a modified PN controller is presented in Section VI.

Given an inadmissible constraint $l^T\mu_p \leq b$, a permissive control law for the enforcement of this constraint can be synthesized using the following steps.

1) Find all inequalities $l'^T\mu_p \leq b'$ that are:
   a) valid transformations of $l^T\mu \leq b$ according to Lemma 9;
   b) admissible constraints according to the theory developed in Section V-A.

   An infinite number of inequalities may exist that meet these two requirements, but they may be expressed with a finite number of inequalities because linearly dependent constraints do not represent different restrictions on the behavior of the plant. Detailed instructions for carrying out this step can be found in [16].
2) Construct the controller incidence matrices associated with these constraints using $D_c = -l'^TD_p$.
3) Enforce the *union* of the individual control laws by following the procedure of Section VI.

The procedure above is similar to the idea of the *supremal controllable sublanguage* [22], [28] from the supervisory control literature. In both cases, all of the valid behaviors of the plant are characterized based on the plant's structure and the desired constrained behavior, and the supervisor is then used to ensure that the behavior of the plant is limited to this set of admissible behaviors.

To say that the procedure above will always result in a maximally permissive control law, the following two points would have to be proven.

1) Maximally permissive control law associated with a plant and constraint $l^T\mu_p \leq b$ can always be expressed as the disjunction of other linear state inequalities.
2) Transformation procedure in Lemma 9 covers all valid constraint transformations; i.e., if for all $\mu_p \geq 0$ such that $l'^T\mu_p \leq b'$, $l\mu_p \leq b$ is also true, then (l', b') can be expressed as a linear function of (l, b) according to the rules and assumptions of Lemma 9.

Li and Wonham [13] have shown that condition 1) is true when the uncontrollable portions of a plant have a certain "tree structure" (see [13]). For the general case, however, the answer is not known.

VI. Enforcing Disjunctions of Linear Constraints

A. Description of Method

The inequality

$$L\mu_p \leq b \tag{32}$$

represents the logical intersection, or conjunction, of n_c separate linear inequalities. That is, if l_i is the ith row of L and b_i is the ith element of b, then (32) is equivalent to

$$\bigwedge_{i=1}^{n_c} l_i^T\mu_p \leq b_i.$$

The feasible solutions to the inequalities form a convex region [6], and the behavior of a Petri net can be restricted to this region by adding further Petri net structures to the net, as was shown in

Section III. A logical union, or disjunction, of linear constraints is, in general, nonconvex and cannot be enforced with maximal permissivity on a Petri net through the use of other Petri net structures because of the linear nature of reachable Petri net state spaces. A proof of this claim appears in [15].

This section will show how a slight modification to the evolution rules of the controller net can be made such that it will act as a maximally permissive supervisor for a class of nonconvex constraints.

The following disjunction of linear inequalities is to be enforced on the marking of a plant with initial marking $\mu_{p_0} \in \mathbb{Z}^n$ (all elements nonnegative) and incidence matrix $D_p \in \mathbb{Z}^{n \times m}$

$$\bigvee_{i=1}^{n_c} l_i^T \mu_p \leq b_i \tag{33}$$

where $l_i \in \mathbb{Z}^n$ and $b_i \in \mathbb{Z}$. Let

$$D_{c_i} = -l_i^T D_p \tag{34}$$

$$\mu_{c_{i0}} = b_i - l_i^T \mu_{p0} \tag{35}$$

for $1 \leq i \leq n_c$. This procedure is the same as detailed in Section III; thus, each pair (D_{c_i}, μ_{c_i}) is a maximally permissive Petri net supervisor for enforcing the constraint $l_i^T \mu_p \leq b_i$. If all of these supervisor elements, however, were to be simultaneously enforced on the plant, then the result would be the logical intersection of the constraints instead of their union.

In order for the controller to enforce a disjunction of inequalities, at least one of the inequalities $l_i^T \mu_p \leq b_i$ must be true at every transition firing iteration of the net's evolution. Let

$$L = [l_1 \quad l_2 \quad \cdots \quad l_{n_c}]^T$$

so that

$$D_c = -LD_p \tag{36}$$

and

$$\mu_{c_0} = b - L\mu_{p_0} \tag{37}$$

which is identical to the controller construction from Section III. The enabling rule for the controller portion of the net, however, must be changed such that it ensures that at least one of the inequalities is being obeyed at all times instead of all of them at all times.

A firing vector q is valid (indicates the firing of an enabled transition) iff

$$D_p^- q \leq \mu_p \tag{38}$$

and

$$\mu_{c_i} + D_{c_i} q \geq 0 \qquad \text{for some } i \in 1 \cdots n_c. \tag{39}$$

Inequality (38) is the standard Petri net enabling condition for a plant that may include transitions with self loops. The enabling condition for the controller (39) does not include any $D_{c_i}^-$ terms because controllers constructed according to the rules in Section III do not contain self loops. Condition (39) may also be written as

$$\max_{i=1}^{n_c} (\mu_{c_i} + D_{c_i} q) \geq 0. \tag{40}$$

Note that it is still true that

$$L\mu_p + \mu_c = b. \tag{41}$$

However, unlike the standard nonnegative slack variables from before, many of the elements of this μ_c may be negative. The restriction placed by condition (40) ensures that at least one of the elements is nonnegative, and thus at any time, at least one of the inequalities in (33) is being satisfied.

Proposition 12—Maximal Permissivity of Disjunction-Enforcing Controllers: A controller constructed according to (36) and (37) using enabling rule (40) is a maximally permissive supervisor for the enforcement of constraint (33) on the plant (D_p, μ_{c_0}) iff

$$\mu_{c_{i_0}} \geq 0 \qquad \text{for some } i \in 1 \cdots n_c. \tag{42}$$

Proof: If condition (42) is not met, then the initial conditions of the plant violate the constraint (33) according to (37), and the constraint cannot be enforced.

Equation (41) shows that the state space of the closed-loop system being outside the bounds of constraint (33) is equivalent to the situation when all elements of μ_c are negative. This is the only condition, however, that is prevented by enabling rule (40). The only time the controller will intervene to disable a transition is when the firing of that transition would cause a direct violation of constraint (33), and thus, the supervisor is maximally permissive. □

Remark: The simple rules that govern ordinary Petri net behavior are what help to make the Petri net model so attractive both for analysis and implementation. The reluctance to modify this model for the enforcement of nonconvex constraints on Petri net plants is overcome for the following reasons.

1) Ability to handle the disjunction of linear constraints as well as their conjunction is a powerful advancement in the utility of the method and is necessary for the proper solution of problems in many applications.
2) Disjunctions of linear constraints are important for the permissive enforcement of linear constraints under conditions in which transitions may be uncontrollable or unobservable.
3) Modified rules for controller state evolution involve only a slight modification of the ordinary transition-enabling rule. Analysis and implementation are similar to that of ordinary Petri nets.

B. Example—An Uncontrollable Loop Is Added to the Assembly Cell

The robots of the piston rod robotic assembly cell are not 100% reliable. It is possible that the M-1 robot will fail to properly secure a piston cap to its rod. The plant is now augmented with an error recovery loop considered to be under the supervision of an auxiliary controller. The modified structure is shown in Fig. 4. The uncontrollable firing of transition t_{10} indicates that a fault has occurred. Place p_9 is then marked with the number of M-1 robots that have experienced faults and have entered the recovery loop. Tokens in p_{10} represent the combined actions of M-1 and S-380 robots to replace and realign the appropriate

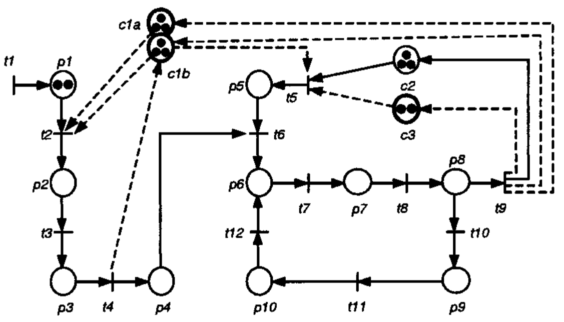

Fig. 4. The final structure of the assembly cell and supervisor. Places c_{1a} and c_{1b} obey the modified PN enabling rule (40) to enforce the nonconvex constraint (45).

parts so that the procedure can begin again at p_6. The new transitions, t_{10}, t_{11}, and t_{12} are all considered uncontrollable and unobservable to the resource managing supervisor.

Constraints (11) and (12) need to be rewritten to account for the use of the two robots in the recovery loop. An S-380 robot is used in p_{10}, and the M-1 robot is required in both p_9 and p_{10}. The new constraints are then

$$\mu_2 + \mu_3 + \mu_{10} \leq 3 \quad \text{(three S-380 robots)} \quad (43)$$
$$\mu_5 + \mu_6 + \mu_7 + \mu_8 + \mu_9 + \mu_{10} \leq 3 \quad \text{(three M-1 robots).} \quad (44)$$

Following the procedure of Section V-B, the kernel, X, of the unobservable incidence matrix, D_{uo}, is computed

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}}_{X} D_{uo} = 0.$$

Admissible constraints will be linearly dependent with the rows of X. The left-hand side of (44) is represented by the fifth row of X; thus the constraint is admissible and requires no transformation. In order to make the left-hand side of (43) an element of the kernel of D_{uo}, we can either add the missing elements from row four or row five of X. Because there is a choice, the transformed constraint will be written as a disjunction of the two candidate inequalities

$$(\mu_2 + \mu_3 + \mu_4 + \mu_6 + \mu_7 + \mu_8 + \mu_9 + \mu_{10} \leq 3)$$
$$\vee (\mu_2 + \mu_3 + \mu_5 + \mu_6 + \mu_7 + \mu_8 + \mu_9 + \mu_{10} \leq 3). \quad (45)$$

A controller is calculated to enforce (45) using the procedure described in Section VI-A. The supervisor, shown with its connections to the plant in Fig. 4, is maximally permissive.

VII. Real-Time Constraint Specifications

Constraints of the form $L\mu_p \leq b$ [inequality (5)] are useful for representing a large variety of forbidden state problems. Finite resource management can be automated using invariant-based control [16], [17] as well as serial, parallel, and general mutual exclusion problems [4], [7]. Both direct and indirect enforcement of constraints on events, i.e., linear inequalities involving the firing vector, can be handled using the invariant-based control method [15], [31]. A class of logical predicates on plant behavior can be transformed into systems of linear inequalities to be enforced by a supervisor [9], [16], [30]. This section explains how the techniques for supervision of ordinary Petri nets can be expanded to timed Petri nets and real-time constraints.

Supervisory controllers are generally driven by sequences of events (firings of transitions) occurring in the plant. Some transitions fire before others; some may fire simultaneously. The states of the plant and controller evolve through time, but no direct representation of time exists. The controller may respond to the firing of transition t_i and then respond to the firing of transition t_j, but no indication exists of how much time has elapsed between the firings of these two transitions. This section will be used to discuss the issues that developed when time is introduced into the control framework.

The most common way of introducing time into a Petri net model of a system is through the use of *timed Petri nets* [23], [25]. A timed net works like an ordinary Petri net, but includes a new function defined on either the transitions or the places of the net. The function indicates the amount of time required for particular transitions to fire or the amount of time that must elapse between the arrival of a token in a place and when it is allowed to take part in enabling and firing another transition. In many cases, the function is simply a constant vector that indicates the timing requirements for each of the net's transitions or places, but it may also be complicated with the firing times relying on the state of the net, the current time, and other factors.

Timed nets are a useful extension of ordinary nets because they do not alter the basic behavior of these nets, they simply provide more information about their evolution, which means that the standard Petri net definitions, including structural invariants, still hold true. A controller that enforces certain state constraints and sequential behaviors on an ordinary Petri net will enforce these same behaviors on the net after timing information is added to it.

The invariant-based control method is implemented through new places and arcs that connect to existing plant transitions. If the timing information of the plant net were associated with the transitions, then the control method could be used without any changes in the method itself. Because the controller has no new transitions of its own, it is not necessary to establish any new timing properties for the controller. It will react to the firings of the plant and will evolve naturally using the plant's own timing. Control goals, such as mutual exclusions, deadlock avoidance, regulation of finite resources, or avoiding forbidden states, may be implemented on timed Petri nets using the method exactly as described. When the elapse of time is associated with transition firings, the resulting behavior of the controlled plant may not be entirely what the control designer expected. Consider the simple constraint

$$\mu_1 + q_1 \leq 1.$$

In standard untimed nets, transition firings are considered to be instantaneous. Because of this, the constraint above means two things: 1) place 1 may never have more than 1 token and 2) transition 1 may not fire if p_1 contains a token. Now, consider if this constraint were to be placed on a Petri net that contained timed transitions. The constraint would take on a third meaning:

3) place 1 may not contain any tokens *while transition 1 is in the process of firing*. In some cases, this result may be exactly what the designer wants; however, the designer must be aware of these subtle changes in the meaning of the constraint inequalities when designing the system.

It is possible to maintain the original meaning of the constraint inequalities by using nets that place their timing information on the places instead of the transitions. Transitions undergo instantaneous firing in these nets, just like in ordinary Petri nets. Constraints may be enforced on nets with timed places using the supervisory control techniques of this paper. The controller will add places to the plant net, and because this is a timed net, timing information must be associated with these new controller places. The tokens in a controller place are used to keep track of the plant's state. They are the controller's bookkeeping device and do not represent a process that requires lengthy amounts of time compared with the time associated with the evolution of the plant. For this reason, *the time delay associated with controller places is defined as zero.*

Difficulties may develop when supervisory control specifications deal directly with time instead of events. Because the controller has no direct access to a clock, it is not possible to directly realize constraints that reference absolute time or relative timing offsets. Some of these kinds of situations may be tackled by including Petri net structures in the plant that the controller can use as an interface between its event-driven world and the world of continuous time. *Clock* and *timer* structures are described in the example below.

Example: Consider a plant with five places and seven transitions and the following two constraints.

1) Transition t_6 may not fire between 6:00 PM and 12:00 AM.
2) Transition t_7 may not fire until at least 2 min has elapsed since the last firing of transition t_4. If t_4 has never fired, then the firing of t_7 is unrestricted.

The supervisory controllers presented here have no access to the current time of day, nor are they informed regarding the amount of time that separates the events to which they react. In order to enforce these constraints, new Petri net structures will be added to the plant, allowing an interface between the controller and time.

A one-way loop of timed Petri net places with a single token can be used to create a clock indicating the current time of day. For example, 24 places connected in a ring, with a 1-h delay for each place, can be used to indicate the current hour of the day. It is not necessary that the delay in every place of the clock be equal, but *the firing rules for the clock must indicate that enabled transitions fire instantly.* Item 1 above indicates that we are concerned with the 6-h period between 6:00 PM and 12:00 AM. The controller will gain access to the current time through the use of the two-place clock shown in Fig. 5(a). The time delays for the two places are

$$\text{Times for} \begin{bmatrix} p_6 \\ p_7 \end{bmatrix} = \begin{bmatrix} 18 \\ 6 \end{bmatrix} \text{hours.}$$

The clock is initialized with a single token in p_6 at 12:00 AM. After 18 h, the token in p_6 will be available to fire t_8. The token in p_6 will transfer to p_7 at 6:00 PM, and it will remain there until

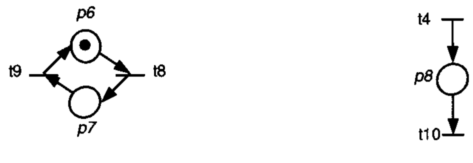

Fig. 5. (a) A two-place time-of-day clock. (b) Place 8 contains tokens whenever fewer than 2 min has elapsed since the last firing of t_4.

midnight. Thus, constraint 1 can be enforced with the inequality $\mu_7 + q_6 \leq 1$.

Constraint 2 does not involve the time of day; rather, it involves a relative offset in time after the firing of transition 4. Transition 7 must wait at least 2 min after the last firing of t_4 before it is allowed to fire. A structure is added to the plant to indicate when less than 2 min has elapsed after the last firing of t_4, which can be done using the net shown in Fig. 5(b), where the time delay for p_8 is 2 min. As with the clock of Fig. 5, the firing rules must indicate that t_{10} fires whenever it is enabled.

Using the net of Fig. 5(b), we know that transition 7 should never be allowed to fire when place 8 contains tokens. An initial guess might suggest that we then enforce the constraint $\mu_8 + q_7 \leq 1$, but item 2 says nothing about limiting the ability of t_4 to fire, and this constraint would prohibit p_8 from ever containing more than one token, indirectly inhibiting the freedom of t_4. Let M be the maximum number of tokens that could ever appear in place 8. This number could be determined through temporal analysis of the plant, or the designer may simply choose M as a number so ridiculously large that t_4 could never fire that many times in a reasonable amount of time. Using this value, the following constraint is then placed on the plant:

$$\mu_8 + Mq_7 \leq M. \tag{46}$$

This constraint will ensure that t_7 never fires if place 8 contains any tokens. If M is too small, then transition t_4 may be indirectly inhibited by constraint (46), so it is necessary to determine a large enough value of M to avoid this situation.

In summary, the control designer should be aware of the following points when introducing timing requirements into the invariant-based control method.

- Defining time delays on the places of a net, instead of its transitions, avoids certain ambiguities in the meanings of combined state/event constraints. Either method may be used, but the meanings of the constraints in each case must be understood.
- When time is defined on the places of a net, time delays associated with the controls are defined as zero to avoid artificial delays in the evolution of the plant.
- Standard supervisory control constraints dealing with the plant state or mutual exclusions can be implemented seamlessly on place-timed Petri net plants.
- External clock and timer structures with appropriate time delays may be added to the plant to provide an interface between the event sequences of supervisory control and real time.
- The accuracy of clocks and timers is ensured by firing rules that insist on the instantaneous firing of enabled transitions for these structures.

- Upper bounds, such as M in the example, can be used to avoid unwanted consequences of state/event related constraints that deal with the conditions of timers and clocks.

VIII. Conclusions

Petri nets possess many assets as models for DES. Concurrent processes and events can be easily modeled within the framework. They provide for larger reachable state spaces, more compact representation, and increased behavioral complexity compared with automata-based models. The goal of this paper has been to present an approach to Petri net supervisory control that is unified and tractable as well as comprehensive and practical.

The primary technical tools required for the use and analysis of the control methods presented here involve Petri net theory and matrix algebra. The main synthesis technique is based on the idea that specifications representing desired plant behavior can be enforced by making them invariants of the closed-loop system. Most of the other tools in this paper also revolve around the creation or characterization of invariants or an analysis of the interrelation between control specifications and plant and controller structure.

Because an invariant-based controller is itself a Petri net, the unified plant/controller system facilitates the use of established synthesis and analysis methods. The closed-loop system can be designed, analyzed, simulated, verified, and augmented using tools already established for Petri nets.

Unobservable transitions have received little attention in the DES Petri net control literature, but they present an important problem, and systems that incorporate unobservable events are of practical concern. Here, the problem of unobservability has been presented and analyzed concurrently and analogously to uncontrollability.

A method has been described for characterizing all feasible invariant-based controllers for enforcing a linear constraint on a plant with uncontrollable and unobservable transitions. This characterization can be combined with an extended Petri net controller definition to enforce the logical union of all these feasible controls. Supervisors designed this way will have a high degree of permissivity. Unfortunately, it has not been demonstrated that these controllers will always be maximally permissive, because it is not known if a situation exists in which the maximally permissive control law corresponding to a linear predicate is ever something other than a disjunction of other linear predicates.

A wide variety of supervisory control goals can be handled with the proposed method. Inequality (5) is not only appropriate for formulating a large range of forbidden state problems, generalized mutual exclusion constraints, and finite resource management problems, but it is also appropriate for specifying a number of supervision goals that are not normally thought of as state-based constraints. The addition of timer and clock structures to a timed Petri net plant model will permit real-time–based constraints to be represented in the form of (5). A number of approaches to deadlock avoidance and ensured liveness can also be reduced to the enforcement of linear inequalities. The integration of these techniques with the invariant-based control method is a current research topic.

Computational efficiency is one of the goals of the supervision techniques presented here. An invariant-based controller is computed very efficiently by a single matrix multiplication, and its size grows polynomially with the number of specifications. Because the controller is a Petri net, control actions are also simple to compute online.

Handling uncontrollable and unobservable transitions does not add any complexity to the online computation of control actions. The increased complexity is encountered only in the initial control design. Computationally tractable techniques have been presented for this process involving the solution of an integer linear program or through the triangularization of integer matrices through constrained row operations (see the Appendix).

Invariant-based supervisors are viable models for real-time control implementations. The speed and efficiency with which they are computed also makes them appropriate for online control reconfiguration because of sensor or actuator faults, or dynamically modified system specifications.

Appendix

A. Generating Constraint Transformations

1) An Integer Linear Program for Calculating R_1 and R_2: The conditions of Theorem 11 can be converted into an integer linear programming problem (ILP) in the standard form. We will consider only a single constraint on the system; multiple constraints can be handled individually and independently. Thus, $n_c = 1$, L and R_1 are vectors, and b and R_2 are scalars.

Let

$$R = [R_1 \quad R_2' \quad R_3]$$

where $R_2' = R_2 - 1$ and $R_3 \in \mathbb{Z}^{n_{uc}}$ is a vector of slack variables. The ILP is defined as

$$\min_R \left(z(R) = R \begin{bmatrix} \mu_{p_0} \\ L\mu_{p_0} - b - 1 \\ 0 \end{bmatrix} \right)$$

$$\text{s.t.} \begin{cases} R \begin{bmatrix} D_{uc} & D_{uo} \\ LD_{uc} & LD_{uo} \\ I & 0 \end{bmatrix} = -L[D_{uc} \quad D_{uo}] \\ R \geq 0 \text{ (integer).} \end{cases} \tag{47}$$

After solving (47), if the minimum of the objective function $z^* = z(R^*)$ is greater than $b - L\mu_{p_0}$, then the problem cannot be solved, as no values of R_1 and R_2 exist that will satisfy the condition in Lemma 10. If the minimum is less than or equal to $b - L\mu_{p_0}$, then transform R_2' back into R_2 and generate the controller using the formulas in Theorem 11.

Remark: Minor difficulties may be encountered when using this method of generating R_1 and R_2. For a controller to exist, we need the objective function of the ILP, $z(R) \leq b - L\mu_{p_0}$; however, it is not clear that we should attempt to minimize this function. Oftentimes, this function will have an unbounded minimum, making it necessary for the designer to introduce an additional constraint to achieve a bounded solution. Care must also be taken such that the ILP does not yield the pathological transformation $L' = R_1 + R_2 L = 0$, when other nonzero possibilities exist for L'.

2) Matrix Row Operations: It is possible to obtain appropriate constraint transformations by performing row operations on a matrix containing the uncontrollable and unobservable columns of the plant incidence matrix. The computational

```
Algorithm 1 (Constraint Transformation).
Input:  $L \in \mathbb{Z}^{n_c \times n}, b \in \mathbb{Z}^{n_c}, D_{uc} \in \mathbb{Z}^{n \times n_{uc}}, D_{uo} \in \mathbb{Z}^{n \times n_{uo}}, \mu_{p_0} \in \mathbb{Z}^n$
if ($LD_{uc} \leq 0$ and $LD_{uo} = 0$) then
    $R_1 := 0_{n_c \times n}, R_2 := I_{n_c \times n_c}$
else
    $M := \begin{bmatrix} D_{uc} & D_{uo} & I \\ LD_{uc} & LD_{uo} & \end{bmatrix}$
    Let $M(i,j)$ represent the $(i,j)^{\text{th}}$ element of matrix $M$.
    Zero all positive elements in the $LD_{uc}$ portion of $M$
      following the procedure in Algorithm 2.
    if $M(n+1 \ldots n+n_c, 1 \ldots n_{uc})$ has any positive elements then
        FAILURE: Controller arcs were introduced by the row
          operations into the uncontrollable portion of the
          plant, and they can not be eliminated due to lack
          of negative elements in $D_{uc}$.
    end if
    Zero all elements in the $LD_{uo}$ portion of the $M$ matrix
      following the procedure in Algorithm 3.
    if $M(n+1 \ldots n+n_c, n_{uc}+1 \ldots n_{uc}+n_{uo})$ has any nonzero
      elements then
        FAILURE: The unobservable portion of the plant
          contains arcs which can not be eliminated.
    end if
    $R_1 := M(n+1 \ldots n+n_c, n_{uc}+n_{uo}+1 \ldots n_{uc}+n_{uo}+n)$
    $R_2 := M(n+1 \ldots n+n_c, n_{uc}+n_{uo}+n+1 \ldots n_{uc}+n_{uo}+n+n_c)$
end if
$L' := R_1 + R_2 L$
$b' := R_2(b+1) - 1$
if $L'\mu_{p_0} > b'$ then
    FAILURE: Control law is infeasible.
end if
Output:  $L'$ and $b'$.
```

Fig. 6.

```
Algorithm 2 (Elimination of positive elements from $D_{uc}$).
Input:  Working matrix $M \in \mathbb{Z}^{(n+n_c) \times (n_{uc}+n_{uo}+n+n_c)}$
$i := 1$
while $i \leq \min(n_{uc}, n)$
    if $M(i \ldots n, i)$ has any negative elements then
        Let $j$ be the index of a row in $M(i \ldots n, i)$ which
          contains a negative element.
        Exchange rows $i$ and $j$ of $M$
        Use the negative pivot value at $M(i,i)$ to eliminate
          all positive integers in the column $M(i \ldots n+n_c, i)$
          (See Algorithm 4.)
    else if $M(n+1 \ldots n+n_c, i)$ has any positive elements then
        FAILURE: A controller arc can not be eliminated
          because there are no negative elements in the
          corresponding column of $D_{uc}$.
    end if
    $i := i+1$
end while
Output $M$ and $i$
```

Fig. 7.

part of this procedure involves little more than the integer triangularization of a matrix, and thus, it is simpler to compute R_1 and R_2 using this method than by using the ILP presented in the previous section. When using the algorithm, recall that $D_{uc} \in \mathbb{Z}^{n \times n_{uc}}$ corresponds to the uncontrollable transitions in the plant *that may be observed*, while all unobservable transitions are represented in $D_{uo} \in \mathbb{Z}^{n \times n_{uo}}$.

As was done in Section I-A, we shall ensure that condition (19) is met by making $R_1 \geq 0$. In terms of row operations, this

```
Algorithm 3 (Zeroing of all elements in $D_{uo}$).
Input:  Working matrix $M \in \mathbb{Z}^{(n+n_c) \times (n_{uc}+n_{uo}+n+n_c)}$ and $i$
while $i \leq \min(n_{uc}+n_{uo}, n)$
    if $M(i \ldots n, i)$ contains any negative elements then
        Let $j$ be the index of a row in $M(i \ldots n, i)$ which
          contains a negative element.
        Exchange rows $i$ and $j$ of $M$
        $k := 1$
    else
        $k := 0$
    end if
    if $M(i \ldots n, i)$ has any positive elements then
        Let $j$ be the index of a row in $M(i \ldots n, i)$ which
          contains a positive element.
        Exchange rows $i+k$ and $j$ of $M$
        Use the positive pivot value at $M(i+k, i)$ to eliminate
          all negative integers in the column
          $M(i+k \ldots n+n_c, i)$ (See Algorithm 4.)
    end if
    if $k = 1$ then
        Use the negative pivot value at $M(i,i)$ to eliminate
          all positive integers in the column $M(i \ldots n+n_c, i)$
    end if
    if $M(n+1 \ldots n+n_c, i)$ has nonzero elements then
        FAILURE: Controller arc(s) can not be eliminated from
          the unobservable portion of the plant.
    end if
    $i := i+1$
end while
Output $M$
```

Fig. 8.

```
Algorithm 4 (Column Zeroing).
Input:  Working matrix $M \in \mathbb{Z}^{(n+n_c) \times (n_{uc}+n_{uo}+n+n_c)}$ and pivot
  position $(p, j)$.
$i := p+1$
while $i \leq n+n_c$
    if $M(i,j)$ is opposite $M(p,j)$ in sign then
        while $M(i,j) \neq 0$
            if $|M(p,j)| > |M(i,j)|$ then
                $d :=$ floor $(-M(p,j)/M(i,j))$
                if ($\text{mod}(M(p,j), M(i,j)) = 0$) then
                    $d := d - 1$ (Don't zero-out the pivot!)
                end if
                $M(p,.) := M(p,.) + dM(i,.)$ (row operation)
            else
                $d :=$ floor $(-M(i,j)/M(p,j))$
                $M(i,.) := M(i,.) + dM(p,.)$ (row operation)
            end if
        end while
    end if
    $i := i+1$
end while
Output $M$
```

Fig. 9.

means that elements in rows are eliminated strictly through addition, never through subtraction, and that rows can be premultiplied only by positive integers. Algorithm 1 in Fig. 6 presents the procedure for determining R_1 and R_2. The procedure for zeroing out the elements in a column of numbers that have the opposite sign of the "pivot" is given in Algorithm 4 in Fig. 9.

Algorithm 1 ensures that condition (20) is met because the procedure for choosing the "pivot" elements never picks from the LD_{uc} and LD_{uo} portions of the M matrix. Combined with the zeroing procedure of Algorithm 4, these steps ensure that the R_2 portion of the M matrix is diagonal with strictly positive elements.

Algorithms 2 and 3 in Figs. 7 and 8, respectively, (called by Algorithm 1) are used to make sure that the transformed constraints meet conditions (17) and (18). The feasibility check at the end of Algorithm 1 directly tests the condition of Lemma 10 to ensure that the controller does exist. The instructions for picking positive or negative elements to act as pivots in the two main loops are left specifically vague. Different methods of choosing the pivot will lead to different constraint transformations. *It would be possible, for instance, to find a basis for all valid constraint transformations by repeating the procedures in Algorithm 1 whenever a choice of more than one pivot for a given column existed.*

References

[1] K. Barkaoui and I. B. Abdallah, "Deadlock avoidance in FMS based on structural theory of petri nets," in *IEEE Symp. Emerging Technol. Factory Automat.*, vol. 2, Piscataway, NJ, 1995, pp. 499–510.

[2] S. Bogdan and F. L. Lewis, "Matrix approach to deadlock avoidance of dispatching in multi-class finite buffer reentrant flow lines," in *Proc. 1997 IEEE Int. Symp. Intell. Contr.*, Piscataway, NJ, July 1997, pp. 397–402.

[3] J. Desel and J. Esparza, *Free Choice Petri Nets*. Cambridge, U.K.: Cambridge Univ. Press, 1995.

[4] A. A. Desrochers and R. Y. Al-Jaar, *Applications of Petri Nets in Manufacturing Systems*. Piscataway, NJ: IEEE Press, 1995.

[5] J. Ezpeleta, J. M. Colom, and J. Martínez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Trans. Robot. Automat.*, vol. 11, pp. 173–184, Apr. 1995.

[6] S.-C. Fang and S. Puthenpura, *Linear Optimization and Extensions: Theory and Algorithms*. Engelwood Cliffs, NJ: Prentice-Hall, 1993.

[7] A. Giua, F. DiCesare, and M. Silva, "Generalized mutual exclusion constraints on nets with uncontrollable transitions," in *Proc. 1992 IEEE Int. Conf. Syst., Man, Cybern.*, Chicago, IL, Oct. 1992, pp. 974–979.

[8] L. E. Holloway, B. H. Krogh, and A. Giua, "A survey of Petri net methods for controlled discrete event systems," *Discrete Event Dyn. Syst.: Theory Applicat.*, vol. 7, pp. 151–190, Apr. 1997.

[9] J. N. Hooker, "A qualitative approach to logical inference," *Decision Support Syst.*, no. 4, pp. 45–69, 1988.

[10] H.-H. Huang, F. L. Lewis, and D. A. Tacconi, "Deadlock analysis using a new matrix-based controller for reentrant flow line design," in *IECON Proc. Ind. Electron. Conf.*, vol. 1, Los Alamitos, CA, 1996, pp. 463–468.

[11] F. L. Lewis, H. Huang, D. Tacconi, A. Gürel, and O. Pastravanu, "Analysis of deadlocks and circular waits using a matrix model for discrete event systems," Automat. Robot. Res. Inst., The Univ. Texas at Arlington, Ft. Worth, TX, Tech. Rep., Oct. 1995.

[12] Y. Li and W. M. Wonham, "Controllability and observability in the state-feedback control of discrete-event systems," in *Proc. 27th Conf. Decision Contr.*, Austin, TX, Dec. 1988, pp. 203–208.

[13] ——, "Control of vector discrete event systems II—Controller synthesis," *IEEE Trans. Automat. Contr.*, vol. 39, no. 3, pp. 512–530, Mar. 1994.

[14] J. Martinez and M. Silva, "A simple and fast algorithm to obtain all invariants of a generalized Petri net," in *Advances in Petri Nets*. Berlin: Springer-Verlag, 1980, Lecture Notes, Comput. Sci., no. 52.

[15] J. O. Moody, "Petri net supervisors for discrete event systems," Ph.D. dissertation, Dept. Elect. Eng., Univ. Notre Dame, Notre Dame, IN, 1998.

[16] J. O. Moody and P. J. Antsaklis, *Supervisory Control of Discrete Event Systems Using Petri Nets*. Amsterdam, The Netherlands: Kluwer, 1998.

[17] J. O. Moody, P. J. Antsaklis, and M. D. Lemmon, "Automated design of a Petri net feedback controller for a robotic assembly cell," in *Proc. 1995 INRIA/IEEE Symp. Emerging Technol. Factory Automat.*, vol. 2, Paris, France, Oct. 1995, pp. 117–128.

[18] ——, "Feedback Petri net control design in the presence of uncontrollable transitions," in *Proc. 34th IEEE Conf. Decision Contr.*, vol. 1, New Orleans, LA, Dec. 1995, pp. 905–906.

[19] J. O. Moody, K. Yamalidou, M. D. Lemmon, and P. J. Antsaklis, "Feedback control of Petri nets based on place invariants," in *Proc. 33rd IEEE Conf. Decision Contr.*, vol. 3, Lake Buena Vista, FL, Dec. 1994, pp. 3104–3109.

[20] T. Murata, "Petri nets: Properties, analysis, and applications," *Proc. IEEE*, vol. 77, pp. 541–580, 1989.

[21] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*. Engelwood Cliffs, NJ: Prentice-Hall, 1981.

[22] P. J. G. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proc. IEEE*, vol. 77, pp. 81–97, Jan. 1989.

[23] C. V. Ramamoorthy and G. S. Ho, "Performance evaluation of asynchronous concurrent systems using Petri nets," *IEEE Trans. Software Eng.*, vol. SE-6, no. 5, 1980.

[24] W. Reisig, *Petri Nets*. Berlin: Springer-Verlag, 1985.

[25] J. Sifakis, "Performance evaluation of systems using Petri nets," in *Advances in Petri Nets*, W. Brauer, Ed. Berlin: Springer-Verlag, 1979, Lecture Notes, Comput. Sci., no. 84.

[26] R. S. Sreenivas, "On commoner's liveness theorem and supervisory policies that enforce liveness in free-choice Petri nets," *Syst. Contr. Lett.*, vol. 31, pp. 41–48, June 1997.

[27] ——, "On the existence of supervisory policies that enforce liveness in discrete-event dynamic systems modeled by controlled Petri nets," *IEEE Trans. Automat. Contr.*, vol. 42, pp. 928–945, July 1997.

[28] W. M. Wonham and P. J. G. Ramadge, "On the supremal controllable sublanguage of a given language," *SIAM J. Contr. Optim.*, vol. 25, no. 3, pp. 637–659, 1987.

[29] K.-Y. Xing, B.-S. Hu, and H.-X. Chen, "Deadlock avoidance policy for Petri net modeling of flexible manufacturing systems with shared resources," *IEEE Trans. Automat. Contr.*, vol. 41, pp. 289–295, Feb. 1996.

[30] K. Yamalidou and J. C. Kantor, "Modeling and optimal control of discrete-event chemical processes using Petri nets," *Comput. Chem. Eng.*, vol. 15, no. 7, pp. 503–519, 1991.

[31] K. Yamalidou, J. O. Moody, M. D. Lemmon, and P. J. Antsaklis, "Feedback control of Petri nets based on place invariants," *Automatica*, vol. 32, pp. 15–28, Jan. 1996.

John O. Moody (S'96–M'98) is an Advisory Engineer and Research Scientist for Lockheed Martin Federal Systems in Owego, NY. He received the doctorate degree in electrical engineering in 1998 from the University of Notre Dame, where he also earned the B.S. and M.S. degrees. He was awarded an Arthur J. Schmitt fellowship as well as the Center for Applied Mathematics fellowship from the University of Notre Dame. He is coauthor of the book *Supervisory Control of Discrete Event Systems Using Petri Nets* (Kluwer Academic 1998, with P. Antsaklis). He is a member of Eta Kappa Nu, Tau Beta Pi, and the IEEE. His research interests include discrete event and hybrid control systems, neural network construction and architecture, parallel computing, and autonomous, intelligent control systems.

Panos J. Antsaklis (S'74–M'76–SM'86–F'91) is Professor of Electrical Engineering at the University of Notre Dame. He received the undergraduate degree from the National Technical University of Athens (NTUA), Greece, and the M.S. and Ph.D. degrees in electrical engineering from Brown University. He has taught and conducted research at Rice University, Imperial College of the University of London, MIT, NTUA and the Technical University of Crete, Greece.

His research interests are in the area of systems and control, with emphasis on hybrid and discrete event systems, on autonomous, intelligent and learning control systems, on methodologies for reconfigurable control and neural networks. He has authored a number of publications in journals, conference proceedings and books, and he has edited three books: *An Introduction to Intelligent and Autonomous Control* (Kluwer Academic 1993; with K. Passino), *Hybrid Systems II* and *Hybrid Systems IV* (Springer 1995 and 1997; with W. Kohn, A. Nerode and S. Sastry). He has also authored a graduate textbook *Linear Systems* (McGraw-Hill 1997; with A. N. Michel).

He serves on the editorial boards of several journals, he has been the Guest Editor of special issues on neural networks (*IEEE Control Systems Magazine*; 1990 and 1992), on intelligence and learning (*IEEE Control Systems Magazine*; 1995), on hybrid control systems (*IEEE Transactions on Automatic Control* 1998) and on hybrid systems (*Journal of Discrete Event Dynamic Systems*, 1998). He has served as Program Chair and General Chair of major systems and control conferences and he was the 1997 President of the IEEE Control Systems Society (CSS). He is an IEEE Fellow.