A Colored Petri Net-Based Formal Method for the Design of Control Systems

M. Makungu, R. St-Denis, and M. Barbeau
Département de mathématiques et d'informatique
Université de Sherbrooke
Sherbrooke (Québec) CANADA J1K 2R1
Email: {makungu, stdenis, barbeau}@dmi.usherb.ca

Abstract

Several formal methods model reactive systems as discrete-event systems (DES). This makes mathematical reasoning about their properties easier and controller synthesis possible. In this paper, we investigate the forbidden state control problem in which a DES is represented as a colored Petri net with a symmetry specification. More specifically, we provide an efficient formal method for synthesizing a controller which, when combined with the original system, will avoid reaching forbidden states. This problem is decidable if the colored Petri net has finite color sets and bounded places. Unlike conventional methods that explore the entire reachable set of states, our method avoids an exhaustive search of the state space by exploiting a symmetry specification. Furthermore, this abstraction technique allows a compact representation for the controller. Therefore, our method performs particularly well when applied to large but structured processes with similar components.

1 Introduction

A discrete-event system (DES) is a dynamic system whose the internal state changes instantaneously in response to the occurrence of an event. Typically, they are used to represent the behavior of reactive, communication, or manufacturing systems. The control theory for discrete-event systems pioneered by Ramadge and Wonham [15] is a framework for modeling supervised discrete-event systems and applying synthesis algorithms to solve control problems. This theory has been primarily studied in the context of automaton-based models. Automaton-based modeling is, however, cumbersome, particularly in representing large systems consisting of numerous similar interacting components. Even though the computational complexity can be polynomial in the number of system states, it grows exponentially with the number of components. This phenomenon, called the *state explosion problem* [1], can be overcomed by partitioning the components into a small number of equivalence classes so that all components in a given class are essentially similar. This paper makes a contribution in this direction and presents an algorithm that reduces the complexity of the supervisory control problem under full observation for instances consisting of large but structured processes with similar behaviors.

Recently, Petri nets have begun to receive attention as models for investigating the control of discrete-event systems. The Petri net-based models are more powerful than the automaton-based models in that the set of Petri net languages is a superset of regular languages. Furthermore, they allow more compact representations of multiple interacting components than equivalent automaton models. Indeed, a single Petri net structure can describe the behavior of several concurrent components modeled as tokens marking the structure.

One may identify mainly two different approaches. In the first approach, introduced by Krogh [10] and Ichikawa [7], a DES is described by using a *Controlled Petri net* (CtlP-net). A CtlP-net is an extension of a standard Petri net in which the enabling of transitions can be influenced by external binary control inputs. A tutorial survey of works related to this approach can be found in [4]. In contrast, the second approach does not include the concept of control input. A DES is specified by using an ordinary Petri net and its dynamic is viewed with a linear algebraic perspective [2] [3] [11].

Among the Petri net-based models, colored Petri nets (CP-nets) [8] are powerful enough to describe complex systems in a manageable way, particularly when they contain many interacting components that are similar but not identical. It is well-known that the class of CP-nets with finite color sets is equivalent to the class of place/transition nets (PT-nets). The CP-

0730-3157/96 $5.00

nets describe the same systems as the PT-nets but often in a more compact form and allow the use of equivalence classes to represent similar components. Specifications can be more readable and eventually more tractable. If we do not impose any restrictions on the definition of CP-nets, then they are equivalent to Turing machines.

The work presented in this paper addresses the forbidden state control problem for a class of CP-nets in which color sets are finite and behaviors satisfy boundedness properties. Given a control specification expressed as a set of forbidden markings, a procedure computes the unique maximal set of admissible markings and a controller by means of an occurrence graph with symmetries. The symmetries guarantee that equivalent markings have similar behavior [9]. Our aim is not to include, as control logic, additional places and transitions to the CP-net being controlled. Such an approach is generally used when a simulation or a performance analysis is performed for a given control policy [10]. In this work, the process being controlled is separated from the control logic. This approach is suitable to controller synthesis in which the process to be controlled already exists and the aim is to compute, from a control specification, a separated controller which, when embodied with the process in a closed-loop system, satisfies the control specification. Thus, this approach permits formulations and solutions of different control synthesis problems for the same process.

The layout of the paper is as follows. The next section introduces the notation and definitions used in CP-nets. Section 3 extends the supervisory control theory to processes modeled by CP-nets with a consistent symmetry specification. Section 4 formulates the control synthesis problem for avoiding a set of forbidden markings. Section 5 describes in detail a new synthesis algorithm, illustrates its application with an example, and outlines a proof of its correctness. Finally, concluding remarks are provided in Section 6.

2 Notation and Preliminaries

We use a CP-net to model a discrete-event system (DES). A colored Petri net is an ordered tuple: $CPN = (\Sigma, P, T, A, N, C, E, M_0)$, where Σ is a finite set of non-empty types, called *color sets*; P is a finite set of *places*; T is a finite set of *transitions*; A is a finite set of *arcs* connecting places and transitions; N is a *node function* that maps each arc into a pair of nodes of different kinds (i.e., one is a place, while the other is a transition); C is a *color function* that associates a color set with each place; E is an *arc expression function* that maps each arc into a multi-set over the color set that is attached to the corresponding place; and M_0 is the initial marking. We consider only CP-nets with decidable properties as valid models for DESs; that is, the color sets are finite and the contents of places are bounded. Furthermore, for the sake of simplicity, we assume that guards (occasionally used in the definition of CP-nets) always evaluate to true.

The following notation is introduced to consider the behavior of a CP-net that is based on the concepts of *enabling* and *occurrence*. The reader is referred to Jensen's book [8] for a detailed exposition. For all $t \in T$, let $Var(t)$ be the set of *variables* appearing on arcs that have t as source or destination. For all $(p,t) \in P \times T$, let $E(p,t) = \sum_{a \in A(p,t)} E(a)$,[1] where $A(p,t)$ gives the set of arcs from p to t. $E(t,p)$ is defined in a similar fashion.

A *binding* of a transition t is a substitution, noted $b = < v_1 = c_1, \ldots, v_n = c_n >$, that assigns a color c_i to the variable $v_i \in Var(t)$ $(i = 1, \ldots, n)$. A *binding element* is a pair (t,b) where $t \in T$ and $b \in B(t)$, the set of all bindings for t. Let BE be the set of all binding elements.

The state of a CP-net is given by its current *marking* M that maps each place p into a multi-set over $C(p)$. A marking gives the current distribution of tokens on the places. A *step* Y is a function that maps each $t \in T$ into a multi-set over $B(t)$ such that $Y(t)$ is finite for all $t \in T$ and non-empty for at least one $t \in T$. A step Y is *enabled* in a marking M if and only if

$$\sum_{b \in Y(t)} E(p,t) < b > \leq M(p) \text{ for all } p \in P$$

where $E(p,t) < b >$ yields the multi-set of tokens removed from p when t occurs with the binding b. The sets of all markings and steps are denoted by $\mathcal{M}$ and $\mathcal{Y}$, respectively. When a step Y is enabled in a marking M, it may *occur*, changing it to another marking M', defined by:

$$M'(p) = (M(p) - \sum_{b \in Y(t)} E(p,t) < b >) + \sum_{b \in Y(t)} E(t,p) < b > \forall p \in P \quad (1)$$

Moreover, we say that M' is *directly reachable* from M by the *occurrence* of the step Y, which is also denoted by $M[Y> M'$. The concept of *occurring step* is extended to a *finite occurrence sequence* of markings and

[1] The expression $\sum_{s \in S} m(s)s$ represents a multi-set. The non-negative integer $m(s)$ is the number of occurrences of the element s in the multi-set m.

steps as follows: $M_1[Y_1> M_2[Y_2> \ldots M_n[Y_n> M_{n+1}$, where $n \in \mathbb{N}$ and $M_i[Y_i> M_{i+1}$ for all $i \in \{1, \ldots, n\}$. The integer n is the *number of steps* in the sequence. The abbreviated form $M_1[Y_1 Y_2 \ldots Y_n> M_{n+1}$ is also used. A marking M' is *reachable* from a marking M if and only if there exists a finite occurrence sequence having M as start marking and M' as end marking. A marking M is reachable if and only if it is reachable from M_0. The set of markings that are reachable from M is denoted as $M[>$. It is always true that $M \in M[>$ for all markings $M \in \mathcal{M}$. In this paper, we assume that an occurring step contains only one binding element, although a number of binding elements can be concurrently enabled. Therefore, $M[Y> M'$ is written as $M[(t,b)> M'$ and $M_1[Y_1 Y_2 \ldots Y_n> M_{n+1}$ as $M_1[(t_1,b_1)(t_2,b_2)\ldots(t_n,b_n)> M_{n+1}$.

As an example, Fig. 1 shows a CP-net that models a unidirectional circular railway with two trains [8]. The track is divided into six different sections. A section is represented by a place (a circle) and the passage from a section to the adjacent section is modeled by a transition (a rectangle). Places can be marked by token colors. In our example, token values model identities of trains. The declaration "color $Train$ = *with* $Tr1 \mid Tr2$" defines the domain of token values. Every place may contain values from the domain $Train$. A token in a given place represents the fact that the corresponding train occupies the corresponding section. Initially, train number 1 ($Tr1$) is in section 1 and train number 2 ($Tr2$) in section 4. The initial marking is represented by the expressions $1`Tr1$ and $1`Tr2$ next to places $S1$ and $S4$ respectively. The integer before the back quote indicates the number of occurrences of the value after the back quote. The execution of a transition always extracts one token x from its incoming place and inserts one token x in its outgoing place. Since the variable x ranges over the color set $Train$, x may only take either value $Tr1$ or $Tr2$. In this model, no constraint is given concerning the position of the two trains relative to each other.

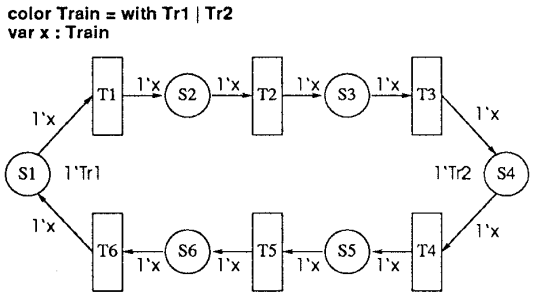

Figure 1: A CP-net describing a circular railway with two trains

It is often the case that large systems have numerous interacting components with similar behaviors. Theses components are so alike that we may abstract the differences between them. This can be captured in a CP-net by defining a set of symmetries called a *symmetry specification* Φ over markings or binding elements [9]. Each symmetry $\phi \in \Phi$ is a function that maps a given marking (or a given binding element) to another marking (or binding element) with similar properties. In order to capture the fact that two symmetrical markings have similar properties, the symmetry specification Φ must be consistent with the behavior of the CP-net. A symmetry specification Φ is *consistent* if and only if the following properties are satisfied for all symmetries $\phi \in \Phi$, all markings $M_1, M_2 \in [M_0>$, and all binding elements $(t,b) \in BE$: (1) $\phi(M_0) = M_0$, and (2) $M_1[(t,b)> M_2 \Leftrightarrow \phi(M_1)[(t,\phi(b))> \phi(M_2)$. A consistent symmetry specification Φ induces the equivalence relations $\approx_M$ and $\approx_{BE}$. The symmetry specification is given by the person who analyses the system. We use $M_{\approx}$ and $BE_{\approx}$ to denote the set of all equivalence classes for $\approx_M$ and $\approx_{BE}$, respectively. The notation $[M]$, where $[M] \in M_{\approx}$, represents all markings equivalent to M. In our example, we use a particular consistent symmetry specification called a permutation [9] and noted Φ_p that allows all possible permutations of the $Train$ color set. Therefore, a marking in which train 1 is in section 1 and train 2 in section 2 is equivalent to a marking in which train 2 is in section 1 and train 1 in section 2.

3 CP-net Supervisory Design Under Full Observation

The basic problem in supervisory control is to construct a controller that can turn off various events of an uncontrolled discrete-event system (DES), called a *process* (Pr), according to some requirements. The process Pr is defined as a triple (CPN, Φ, K), where CPN is a CP-net with a consistent symmetry specification Φ and $K \in [P \rightarrow \mathbb{N} - \{0\}]$ is a *capacity function* bounding the contents of every place. Therefore, we require that $M_0(p) \leq K(p)$ and $M'(p) \leq K(p)$ for all $p \in P$ in the enabling rule (1). In our example, capacity is two.

Let Γ be the set of all functions γ, called *control patterns*, that assign a subset of $B(t)$ to every transition t of T. In our particular example, $B(t) = \{< x = Tr1 >, < x = Tr2 >\}$ for all $t \in T$. If $b \in \gamma(t)$, then the controller prevents the transition t from oc-

curring with the binding b. Let T_c and T_u be fixed disjoint subsets of T denoting the sets of *controllable* and *uncontrollable* transitions, respectively.

A *controlled discrete-event system* (CDES) is an ordered tuple $Pr_c = (CPN_c, \Phi, K, \Gamma)$. In a CPN_c, $M[_c(t,b)> M'$ denotes that a marking M' is directly reachable from M by the occurrence of the binding element (t,b) under the control of $\gamma \in \Gamma$. This is defined as:

$$\begin{array}{ll} M[(t,b)> M' & \text{if } t \in T_u \\ M[(t,b)> M', b \notin \gamma(t) & \text{if } t \in T_c \text{ and } \gamma \in \Gamma \end{array}$$

A *controller* is a pair $S = (G, \varphi)$, where G is a subgraph of an occurrence graph with symmetries (*OS-graph*) and φ the feedback function. An *OS-graph* is a 4-tuple (V, A, N, v_0), where

- V is the finite set of nodes $\{[M] \in M_{\approx} \mid [M] \cap [M_0> \neq \emptyset\}$;
- A is the finite set of arcs $\{([M_1], [(t,b)], [M_2]) \in V \times BE_{\approx} \times V \mid \exists (M_1', (t,b'), M_2') \in [M_1] \times [(t,b)] \times [M_2]$ such that $M_1'[(t,b')> M_2'\}$;
- N is the labeled function that maps each arc a into a representative binding element. If $a = ([M_1], [(t,b)], [M_2]) \in A$, then $N(a) = (t,b)$;
- $v_0 \in V$ is the initial node $(v_0 = [M_0])$.

The *feedback function* $\varphi \in [V \to \Gamma]$ satisfies the following conditions:[2]

$$\begin{array}{ll} \varphi(v)(t) = \emptyset & \text{if } t \in T_u, v \in V \\ \varphi(v)(t) \in \wp(B(t)) & \text{if } t \in T_c, v \in V \end{array}$$

The graph G can be interpreted as the transition graph of an automaton modeling the behavior of the controller as in the original framework of Ramadge and Wonham. It is driven by a sequence of steps occurring in CPN. That is, after the firing of a transition, G moves to a node v which represents the marking reached by CPN. The role of the feedback function φ is to provide, after each execution step of CPN and G, the control pattern γ that represents the binding elements inhibited for the next step.

The CDES and controller are then embodied in a closed-loop system to constitute a *supervised discrete-event system* (SDES) $S/Pr_c = (S, Pr_c)$. A state of a $SDES$ is a pair (v, M) where $v \in V$ and M is a marking of CPN_c such that $M \approx_M M'$ if $v = [M']$.

[2] $\wp(B(t))$ denotes the power set of $B(t)$.

It should be noted that a node v is a representation of markings equivalent to M'. Therefore, v and $[M']$ are interchangeable. The behavior of S/Pr_c is illustrated in Fig. 2. Let M_i be the current marking of the process and (t_i, b_i) the next step such that $M_i[(t_i,b_i)> M_{i+1}$ and $b_i \notin \gamma(t_i)$. Let $v_i = [M_i']$ be a node of G such that $M_i' = \phi(M_i)$. First, the controller S moves to the next node $v_{i+1} = [M_{i+1}']$ by executing the transition from node v_i on the arc labeled (t_i, b_i'). The binding element (t_i, b_i') is the representative member of binding elements equivalent to the process step (t_i, b_i), that is, $(t_i, b_i') = (t_i, \phi(b_i))$. The controller S owns control patterns for representative marking of each equivalence class of reachable markings. Let $\gamma_{i+1}' = \varphi(v_{i+1})$ be the control pattern of node v_{i+1}. To compute the control pattern corresponding to the next marking M_{i+1} of the process, the controller uses the inverse of symmetry function ϕ to map the control pattern γ_{i+1}' into the control pattern $\gamma_{i+1} = \phi^{-1}(\gamma_{i+1}')$.[3] Formally, if (v_i, M_i) is the current state of S/Pr_c, then the next state is (v_{i+1}, M_{i+1}) if and only if there exists a binding element (t_i, b_i) such that $M_i[(t_i,b_i)> M_{i+1}$, $b_i \notin \gamma(t_i)$, and $(v_i, [(t_i, b_i')], v_{i+1}) \in A$ with $(t_i, b_i) \approx_{BE} (t_i, b_i')$. The control pattern of M_{i+1} is $\phi^{-1}(\varphi(v_{n+1}))$. The controller S must always be *complete* in the sense that $(v_i, [(t_i, b_i')], v_{i+1}) \in A$ whenever S/Pr_c is in state (v_i, M_i) with $M_i[_c(t_i,b_i)> M_{i+1}$ and $b_i' \notin \varphi(v_i)(t_i)$.

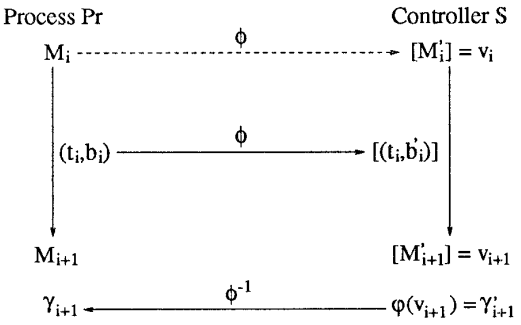

Figure 2: An execution step of the closed-loop system

4 The Forbidden State Control Problem

Several types of control specifications can be applied to control synthesis methods, including: avoiding a set of forbidden states [10]; enforcing event language specifications [15]; and enforcing liveness [6]. In this paper, we consider the forbidden state control problem in which the control specification is expressed as

[3] Since Φ is a group, ϕ^{-1} always exists.

a set of forbidden markings $\mathcal{M}_b$. Based on the consistency property of Φ, if a marking is forbidden, all its equivalent markings are also forbidden. Therefore, only one representative per each equivalence class of forbidden markings is included in $\mathcal{M}_b$. This problem is expressed as follows. Given a set of forbidden markings $\mathcal{M}_b$, an uncontrolled discrete-event system (CPN, Φ, K), and an initial marking $M_0 \notin \mathcal{M}_b$, derive a *maximally permissive* controller S, that is: (1) the closed-loop system S/Pr_c is safe (Pr_c cannot reach a forbidden marking under the control of S); (2) a reachable marking of Pr, which is a nonreachable marking of Pr_c under the control of S, is either forbidden or can uncontrollably lead to a forbidden marking.

Let us consider the forbidden markings for the process described in Fig. 1. The behavior of the trains must be restrained to prevent the trains from colliding. Therefore, the two trains must be separated by at least one section to ensure that an incoming train can stop at a proper distance. It is assumed that transitions $T4$ and $T6$ are uncontrollable. The set of forbidden markings is formally specified as

$$\begin{aligned}\mathcal{M}_b = \{ \quad & \Gamma(S1,Tr1) + \Gamma(S1,Tr2), \ldots, \\ & \Gamma(S6,Tr1) + \Gamma(S6,Tr2), \\ & \Gamma(S1,Tr1) + \Gamma(S2,Tr2), \ldots, \\ & \Gamma(S6,Tr1) + \Gamma(S1,Tr2)\}\end{aligned}$$

As mentioned above, a forbidden marking which has an equivalent marking already in $\mathcal{M}_b$ is not inserted in $\mathcal{M}_b$. For example, forbidden markings $\Gamma(S1,Tr1)+\Gamma(S2,Tr2)$ and $\Gamma(S1,Tr2)+\Gamma(S2,Tr1)$ are equivalents, only one representative is included in $\mathcal{M}_b$. Before describing the synthesis algorithm, lets us introduce an admissibility assessment predicate and the notion of latest controllable binding elements.

4.1 Inadmissible marking

Given a set of forbidden markings $\mathcal{M}_b$, there is, in general, a larger set of markings which must be avoided, due to uncontrollable transition sequences. The markings from which the process can uncontrollably reach forbidden markings are characterized by the recursive predicate *Inadmissible* which has a marking M as argument and returns:

true	if $(\exists t \in T_u)(\exists b \in B(t))\ M[(t,b)\rangle \in [\mathcal{M}_b]$ or if $(\exists t \in T_u)(\exists b \in B(t))$ $Inadmissible(M[(t,b)\rangle)$
false	otherwise

A node with associated inadmissible markings is inadmissible. When the initial marking is safe and steps contain only one binding element, the maximally permissive solution to the forbidden state control problem exists and prevents the process from reaching any inadmissible marking [5].

4.2 Latest Controllable Binding Elements

Let x, y, and z denote nodes in an OS-graph, and (t,b) a binding element, where $t \in T_c$. The *latest controllable binding elements* of z is a set (denoted as $LCBE$) of all triples of the form (x,t,b) such that for each triple:[4]

1. $M_x[(t,b)\rangle\ M_y$
2. $M_y[(t_1,b_1)(t_2,b_2)\ldots(t_n,b_n)\rangle\ M_z$ with $t_i \in T_u$, for $i = 1,\ldots,n$.

A triple (x,t,b) contained in the $LCBE$ of z is interpreted as follows. The occurrence of the step (t,b) from M_x is controllable whereas the sequence of steps $(t_1,b_1)\ldots(t_n,b_n)$ from M_y to M_z are uncontrollable. Therefore, to make unreachable M_z and its equivalents in the process, it is necessary to disable the binding b for t when the process is in marking M_x.

5 The Synthesis Algorithm

The basic idea behind our synthesis algorithm is to reduce the number of markings that must be examined by gathering the components that "behave in the same way" into the same equivalence class. Therefore, the sets of all markings and steps are partitioned into disjoint nonempty equivalence classes. The algorithm, given in Fig. 3, is based on the notion of *latest controllable binding elements*, *equivalent marking*, and *inadmissible marking*. It accepts as input a process Pr, a consistent symmetry specification represented by the equivalence relations $\approx_M$ and $\approx_{BE}$, a set of forbidden markings $\mathcal{M}_b$, and a set of controllable transitions T_c. A maximally permissive compact controller $S = (G, \varphi)$ is computed from representative members, one per class of equivalent markings and class of equivalent steps.

The algorithm uses many functions briefly described hereafter. The function `New_Node` creates a new node from a marking and an LCBE. The function `New_Marking` yields the marking reached after the occurrence of a binding element from a given marking. The function `New_Arc` creates a new arc from a source node, a binding element, and a destination node. The function `From_Nodes` takes as argument a node x and returns the set of nodes in V from which x is directly reachable on an uncontrollable binding element.

[4] The expressions M_x and $lcbe_x$ denote a representative marking M and $LCBE$ of the node x, respectively.

```
1  function Synthesize_Controller(Pr,≈_M,≈_BE,M_b,T_c)
2  V ← {}; A ← {}; processed_nodes ← {};
                   unprocessed_nodes ← {New_Node(M_0,{})}
3  repeat
4    select x in unprocessed_nodes;
5    if not (M_x ≈_M M_y) for some y in processed_nodes then
6      for all (t,b) enabled in M_x do
7        M ← New_Marking(M_x,(t,b));
8        if M ≈_M M' for some M' in M_b then
9          if t ∈ T_c then
10           φ(x)(t) ← φ(x)(t) ∪ {b}
11         else
12           Inadmissible(x); break
13       else
14         if not (M ≈_M M_y) for some y being a son of x then
15           if t ∈ T_c then lcbe_z ← {(x,t,b)} else lcbe_z ← lcbe_x;
16           z ← New_Node(M,lcbe_z);
17           unprocessed_nodes ← unprocessed_nodes ∪ {z};
18           a ← New_Arc(x,(t,b),z); A ← A ∪ a
19     if x.status = admissible then V ← V ∪ {x}
20   else
21     if y.status = inadmissible then Inadmissible(x)
                                  else lcbe_y ← lcbe_y ∪ lcbe_x
22   processed ← processed ∪ {x};
23   unprocessed_nodes ← unprocessed_nodes \ {x}
24 until unprocessed_nodes = {};
```

Figure 3: Algorithm for synthesizing a compact controller

Each node has a status indicating whether or not its associated marking is *admissible*. A new node has its status set to *admissible*. The function **Inadmissible**, given in Fig. 4, fixes the status of a node to *inadmissible* (line 2), determines those that become inadmissible among its predecessors (lines 3 and 4), and updates the feedback function by inserting, for each (x', t, b) in the LCBE of x, the binding b in the set of forbidden bindings for t of x' (lines 6 and 7). When a node becomes inadmissible, all its son nodes in V are removed (including their bound arcs) by using the procedure **Remove_Sons** (line 2). If the LCBE of x is empty, then there are no solutions (line 8).

```
1 procedure Inadmissible(x) :
2     x.status ← inadmissible; Remove_Sons(x); V ← V \ {x}
3     for each x' in From_Nodes(x)
                  such that x'.status = admissible do
4         Inadmissible(x')
5     if lcbe_x ≠ {} then
6         for each (x',t,b) ∈ lcbe_x do
7             φ(x')(t) ← φ(x')(t) ∪ {b}
8     else "no solutions"
9 end
```

Figure 4: Procedure for determining inadmissible nodes

The algorithm works as follows. Initially, the sets of nodes V and arcs A of the graph are both empty. The set of processed nodes is also empty. The node $v_0 = (M_0, \{\})$ is created and inserted into the set of unprocessed nodes (line 2). In the example, $M_0 = \Gamma(S1, Tr1) + \Gamma(S4, Tr2)$ and the $LCBE$ of v_0 is empty because we assume that $M_0 \notin \mathcal{M}_b$. While there are unprocessed nodes, a node x is selected (line 4) and processed. The processing of a node starts with a test for an equivalence between the marking M_x and the marking M_y of an already processed node y (line 5). Only the first-picked node in each equivalence class is developed further. If such a node y exists, then the algorithm checks if it is an inadmissible node. If so, then the **Inadmissible** function is called on node x to disable the latest controllable binding elements on the path leading to x. Otherwise, the contents of $lcbe_x$ is inserted in $lcbe_y$ (line 21). If such a node y does not exist, every binding element (t, b) enabled in M_x is analyzed (line 6). The marking M reached after the occurrence of the binding element (t, b) is computed (line 7) and checked for an equivalence with some other marking M' included in the set of forbidden markings (line 8). If so, there are two cases: either t is controllable or not. If transition t is controllable, the binding b is inserted in the set of forbidden bindings of x (line 10). For example, the marking $\Gamma(S3, Tr1) + \Gamma(S4, Tr2)$, reached after the occurrence of the step $(T2, < x = Tr1 >)$ from node #2 in Fig. 5, is a forbidden marking. Therefore, the binding $< x = Tr1 >$ must be added to $\varphi(\#2)(T2)$. If t is uncontrollable, then the function **Inadmissible** is called on node x (line 12).

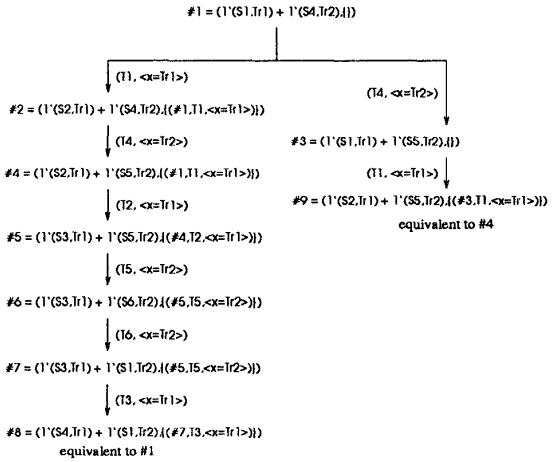

Figure 5: The graph for the circular railway system

Let us consider a part of the graph of Fig. 5 as illustrated in Fig. 6. The marking $\Gamma(S2, Tr1) + \Gamma(S1, Tr2)$, reached after the occurrence of the step $(T6, < x = Tr2 >)$ from node #10, is a forbidden marking. Since the transition $T6$ is uncontrollable, the controller can-

not prevent the step from occurring in the process at this level. Based on the information in $lcbe_{\#10}$, the binding $< x = Tr2 >$ is inserted into $\varphi(\#4)(T5)$.

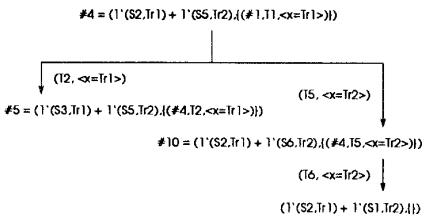

Figure 6: A part of the graph

If M is not equivalent to some forbidden marking in $\mathcal{M}_b$, then the algorithm checks if marking M is equivalent to a marking M_y where y is a son of x (line 14). If all the above conditions are not satisfied, it means that the marking M is not equivalent to a marking of a son of x and not forbidden (for example, node #7 in Fig. 5). In this case, a new node z is created with the following attributes: the marking M and $lcbe_z$ which is defined as $\{(x,t,b)\}$ if the transition t is controllable; otherwise, it is the $lcbe_x$ of its parent node x (lines 15 and 16). Furthermore, the set of unprocessed nodes is updated (line 17) and a new arc is created from node x to node z and added to A (line 18). At the end of the analysis of all binding elements (t,b) enabled in M_x, if the status of x is *admissible*, then x is included in V. Finally, the node x is included in the set of processed nodes and removed from the set of unprocessed nodes (lines 22 and 23).

Markings	T1	T2	T3	T5
$1`(S2,Tr1) + 1`(S4,Tr2)$	$\emptyset$	$Tr1$	$\emptyset$	$\emptyset$
$1`(S2,Tr1) + 1`(S5,Tr2)$	$\emptyset$	$\emptyset$	$\emptyset$	$Tr2$
$1`(S3,Tr1) + 1`(S5,Tr2)$	$\emptyset$	$\emptyset$	$Tr1$	$\emptyset$
$1`(S3,Tr1) + 1`(S6,Tr2)$	$\emptyset$	$\emptyset$	$Tr1$	$\emptyset$
$1`(S3,Tr1) + 1`(S1,Tr2)$	$Tr2$	$\emptyset$	$\emptyset$	$\emptyset$
$1`(S1,Tr1) + 1`(S5,Tr2)$	$\emptyset$	$\emptyset$	$\emptyset$	$Tr2$

Table 1: The feedback function φ

Fig. 5 shows the final solution for the circular-railway example. Less than half of the possible nodes have been inserted in the graph. The feedback function φ is given in Table 1[5]. Since the $LCBE$ of each node is useless in the closed-loop system, only the marking of each node is kept in the solution.

[5]In column labeled T_i, $i = 1,2,3$ or 5, Tr_k must be read $\{< x = Tr_k >\}$ ($k = 1$ or 2).

To show how a discrete-event system is controlled, we define an instantaneous closed-loop system configuration as an ordered tuple $[M, v, \gamma]$, where $M \in \mathcal{M}$, $v \in V$, and $\gamma \in \Gamma$. The notation

$$[M_1, v_1, \gamma_1] \xrightarrow{(t,b)} [M_2, v_2, \gamma_2]$$

indicates that the configuration $[M_2, v_2, \gamma_2]$ is obtained from $[M_1, v_1, \gamma_1]$ by the execution of one step of the closed-loop system. An example of control is given by the following steps:

$$[1`(S4,Tr1) + 1`(S1,Tr2), \#1, \{\}]$$
$$\xrightarrow{(T1,\{x=Tr2\})} [1`(S4,Tr1) + 1`(S2,Tr2), \#2, \{T2 : Tr1\}]$$
$$\xrightarrow{(T4,\{x=Tr1\})} [1`(S5,Tr1) + 1`(S2,Tr2), \#4, \{T5 : Tr2\}]$$
$$\xrightarrow{(T2,\{x=Tr2\})} [1`(S5,Tr1) + 1`(S3,Tr2), \#5, \{T3 : Tr1\}]$$
$$\ldots$$

The correctness of the algorithm is based on the following theorem.

Theorem 1 *The procedure* `Synthesize_Controller` *always terminates and, if* $M_0 \notin [\mathcal{M}_b]$ *and not* $Inadmissible(M_0)$*, then it returns a controller* $S = (G, \varphi)$ *that is the unique maximal solution to the forbidden state control problem with respect to the set of forbidden markings* $\mathcal{M}_b$.

The proof of this theorem is presented in [12]. It is based on the following two lemmas. Lemma 1 reflects some properties of the Jensen's *OS-graph* [9]. Lemma 2 concerns an invariant of the main loop.

Lemma 1 *Let* Φ *be a consistent symmetry specification. For all* $M \in [M_0 >$ *and all* $\phi \in \Phi$, $Inadmissible(M)$ *iff* $Inadmissible(\phi(M))$.

Lemma 2 *Let predicate* $Inadmissible$ *be restricted to markings labeling nodes in the set* `processed_nodes`*. For all* $M \in [M_0>$ *such that there exists a node* v *with* $M_v = M$, $v \in V$ *iff not* $Inadmissible(M_v)$.

6 Summary and Conclusions

In this paper, we have presented a colored Petri-net approach to the control of discrete-event systems. One of the benefits of using CP-nets instead of equivalent PT-nets is the compact and readable representation of the system. The algorithm developed for synthesizing the controller avoids an exhaustive search of the state space by the use of equivalence relations. For systems with many similar components, our algorithm will generally be more efficient than methods requiring the construction of the entire reachable set of states.

The theoretical limitation of our algorithm is, however, due to the restrictions introduced in Section 3.

Comparison given in Table 2 shows that our method is efficient relative to the automaton-based approach. It concerns the same problem as the one introduced in Section 2, but the track is divided in ten sections and the transitions $T1$, $T3$, $T5$, $T7$, and $T9$ are controllable. This table gives the number of states (and transitions) in the process, the control specification, and the controller when automata are used. This table also gives the number of places in the process, the number of forbidden markings in the control specification, and the number of nodes[6] (and arcs) in the graph associated to the controller when the colored Petri net is the formalism used in conjunction with equivalent markings. Note that these controllers have been calculated by a software tool that support both the automaton-based [14] and colored Petri net-based approaches. The reader can find, in a companion paper [13], an application of our method to the congestion management in virtual circuit networks.

Trains	Process		Control specification	
	Automaton	CP-net	Automaton	$\mathcal{M}_b$
2	10^2 (2×10^2)	10	70 (120)	20
3	10^3 (3×10^3)	10	150 (300)	180
4	10^4 (4×10^4)	10	100 (160)	1800

Trains	Controller	
	Automaton	graph
2	60 (100)	30 (50) [54]
3	90 (150)	30 (50) [64]
4	40 (40)	10 (10) [17]

Table 2: Comparison of the automata and CP-nets approaches

Acknowledgments

The research described in this paper was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Fonds pour la formation de chercheurs et l'aide à la recherche (FCAR).

References

[1] E. M. Clarke and O. Grümberg, "Avoiding the state explosion problem in temporal logic model checking algorithms," *Proceedings of the Sixth Annual ACM Symposium on the Principles of Distributed Computing*, 1987, 294-303.

[2] M. J. Denham, "A Petri-net approach to the control of discrete-event systems," in M. J. Denham and A. J. Laub, Eds., *Advanced Computing Concepts and Techniques in Control Engineering*, NATO ASI Series, Vol. F47, Berlin: Springer-Verlag, 1988, 191-214.

[3] A. Giua and F. DiCesare, "Supervisory design using Petri nets," *Proceedings of the 30th Conference on Decision and Control*, 1991, 92-97.

[4] L. E. Holloway and B. H. Krogh, "Controlled Petri nets: A tutorial survey," *Lectures Notes in Control and Information Sciences*, 199, Springer Verlag, London, 1994, 158-168.

[5] L. E. Holloway and B. H. Krogh, "Synthesis of feedback control logic for a class of controlled Petri nets," *IEEE Transactions on Automatic Control*, 35 (5), 1990, 514-523.

[6] L. E. Holloway and X. Guan, "A generalization of state avoidance policies for controlled Petri nets," Technical Report T93002, Technology from the Center for Robotics and Manufacturing Systems, September 1993.

[7] A. Ichikawa and K. Hiraishi, "Analysis and control of discrete event systems represented by Petri nets", *Discrete Event Systems: Models and Applications*, Springer Verlag, New York, 1988.

[8] K. Jensen, *Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use*, Volume 1, Berlin: Springer-Verlag, 1992.

[9] K. Jensen, *Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use*, Volume 2, Berlin: Springer-Verlag, 1995.

[10] B. H. Krogh, "Controlled Petri nets and maximally permissive feedback logic," *Proceedings of the 25th Annual Allerton Conference on Communication, Control, and Computing*, 1987, 317-326.

[11] Y. Li and W. M. Wonham, "Control of vector discrete-event systems I - The base model", *IEEE Transactions on Automatic Control*, 38 (8), 1993, 1214-1227.

[12] M. Makungu, *Formal Methods for Telecommunication Software*, Ph.D. thesis, Université de Sherbrooke, in preparation.

[13] M. Makungu, M. Barbeau, and R. St-Denis, "A supervisory control theoretical approach to congestion management", *Proceedings of ICCT'96*, Beijing, May 1996, to appear.

[14] J. M. Palmier, M. Makungu, F. E. Agapie, M. Barbeau, and R. St-Denis, "An introduction to a synchronized Petri nets tool for the synthesis of discrete event systems", *Proceedings of BMW'94*, Méthodes mathématiques pour la synthèse des systèmes informatiques, Publication no 15, 1994, 67-78, LACIM, UQAM.

[15] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journ ' Control and Optimization*, 25 (1), 1987, 206-230.

[6] The number of nodes that have been generated by the algorithm is also given in brackets.