On the Transformation of Maximally Permissive Marking-based Liveness Enforcing Supervisors into Monitor Supervisors

Kevin X. He and Michael D. Lemmon [1]
Dept. of Electrical Engineering, Univ. of Notre Dame,
Notre Dame, IN 46556, USA
Tel:(219)-631-9691, Fax:(219)-631-4393
xhe, lemmon@maddog.ee.nd.edu

Abstract

A *marking-based* Petri net supervisor restricts a Petri net's behavior by disabling controllable transitions. Monitors [3] are control places that enforce generalized mutual exclusion constraints. A long standing question concerns the relationship between marking-based supervisors and monitors. There are necessary and sufficient conditions for the existence of maximally permissive marking based supervisors enforcing liveness [17], but associated monitor solutions may not be maximally permissive. This paper proves that for a bounded Petri net, a maximally permissive liveness enforcing marking based supervisor can always be transformed into a monitor supervisor. Specific conditions are identified that characterize when the resulting monitor supervisor will or will not be maximally permissive. Several examples are used to illustrate these results.

1 Introduction

An ordinary Petri net is *live* if it is possible to reach any transition from any reachable marking. At each reachable marking, a *marking-based* supervisor disables the firing of a selected set of controllable transitions. The marking based supervisor is said to be *liveness-enforcing*, if its supervisory policy ensures that the controlled Petri net is live. If a liveness-enforcing supervisor exists, then we know that there also exists a *maximally permissive* supervisor. Necessary and sufficient conditions for the existence of marking-based liveness-enforcing supervisors have been established [17, 7]. The determination of such supervisors, however, can be computationally expensive since it invariably involves a search of the Petri net's state space.

Monitors [3] represent another important class of Petri net supervisor. A monitor is a collection of *control places* that enforce generalized mutual exclusion constraints (GMEC) on the network's marking vector. Monitor implementations of supervisors are often considered highly desirable, because the supervisor and plant are both Petri nets, thereby providing a single unified modeling paradigm for controller and plant. In recent years, moreover, extremely efficient methods for computing monitors enforcing linear inequality constraints on the network's marking vector have appeared [13, 14]. There has been some progress in obtaining monitor supervisors [8] enforcing deadlock freedom. In spite of this progress, however, it is difficult to ensure that a monitor supervisor enforcing deadlock freedom exists and if it does exist, it may not be maximally permissive.

A key issue addressed in this paper is the relationship between monitors and marking based supervisors enforcing liveness. In particular, this paper identifies necessary and sufficient conditions under which a monitor implements the maximally permissive liveness enforcing supervisory policy when there is no cyclic lock in the net system. The results are based on recent results [6, 7] in which a partial order method known as network unfolding is used to characterize liveness-enforcing supervisory policies for ordinary Petri nets.

The remainder of this paper is organized as follows. Section 2 reviews definitions and concepts related to the supervisory control of Petri nets. Section 3 summarizes recent results [6, 7] on liveness-enforcing supervision of ordinary Petri nets. Section 4 presents the main results that transform maximally permissive liveness-enforcing marking based supervisors to monitors. Finally, section 5 concludes with directions for future research.

[1]We gratefully acknowledge the partial financial support of the Army Research Office (DAAH04-96-10285, DAAG5-98-1-0199) and the National Science Foundation (NSF-ECS95-31485)

2 Supervisory Control of Petri Nets

This section reviews the definition of ordinary Petri nets and states the supervisory control problem. For more details on Petri nets, refer to [16, 15, 1]. For more details on supervisory control refer to [4, 10, 11].

2.1 Petri nets

An ordinary Petri net $\mathcal{N}$ is represented by the 3-tuple, (S, T, F) where S is the set of *places*, T is the set of *transitions*, $F \subset (S \times T) \cup (T \times S)$ is a set of input arcs (from places to transitions) and output arcs (from transitions to places). The current "state" of the Petri net is represented by the *marking* of the net. The *marking* $\mu : S \to Z^+$ is a map from each place $s \in S$ onto a non-negative integer $\mu(s)$.

The dynamics of a Petri net are described by how the net's marking evolves in response to the *firing* of transitions. A transition $t \in T$ is said to be *enabled* if and only if every place in its preset is marked. An enabled transition may *fire*. If the enabled transition t fires, then the marking of the Petri net is changed by removing one token from every place in t's preset ($\bullet t$) and adding one token to every place to t's postset ($t\bullet$).

We define a *net system* Σ as the pair $(\mathcal{N}, \mu_0)$ where $\mathcal{N} = (S, T, F)$ is an ordinary Petri net and μ_0 is its initial marking. We define $R(\Sigma)$ as the set of all markings reachable from μ_0. We say that the net system is n-safe or bounded if there exists a finite number N such that $\mu(s) \leq N$ for all $s \in S$ and all $\mu \in R(\Sigma)$. The results in this paper pertain to n-safe Petri nets.

Let $\mathcal{N} = (S, T, F)$ be an acyclic net and $x_1, x_2 \in S \cup T$ be two *nodes* of $\mathcal{N}$. We say node x_1 *precedes* node x_2 (denoted as $x_1 < x_2$) if there is a path in the net that goes from x_1 to x_2. We say x_1, x_2 are in *conflict* (denoted as $x_1 \# x_2$) if there exist distinct transitions t_1 and t_2 such that $t_1 < x_1$, $t_2 < x_2$, and $\bullet t_1 \cap \bullet t_2 \neq \emptyset$. We say x_1, x_2 are *concurrent* (denoted as $x_1 || x_2$) if these nodes are not in precedence and not in conflict.

A net system is said to be *live* if for any reachable marking μ and any $t \in T$ there exists a marking μ_t reachable from μ and μ_t enables t.

2.2 Supervisory Control of Petri Nets

We define a *net supervisor* as a mapping $\mathcal{S} : R(\Sigma) \times T \to \{0, 1\}$, where $R(\Sigma)$ is the set of all reachable markings for the net system Σ. A transition t is said to be control-enabled (control-disabled) at a marking μ, if $\mathcal{S}(\mu, t) = 1$ ($\mathcal{S}(\mu, t) = 0$). A *supervised net system* $\Sigma|\mathcal{S}$ is a net system in which only control-enabled transitions can fire.

Consider a net system Σ with accepted language $L(\Sigma)$. We identify a set $R_f(\Sigma)$ of *forbidden markings* which is a subset of $R(\Sigma)$. The language accepted by a supervised net system $L(\Sigma|\mathcal{S})$ is said to be *legal* if there is no occurrence sequence $\sigma \in L(\Sigma|\mathcal{S})$ such that $\mu_0 \xrightarrow{\sigma} \mu_f$ where $\mu_f \in R_f(\Sigma)$. For a given set of forbidden markings $R_f(\Sigma)$, there may be many *legal* controllable sublanguages. The largest such language that contains all other legal controllable sublanguages is called the *supremal controllable sublanguage*. We say $\mathcal{S}$ is *maximally permissive*, if $\Sigma|\mathcal{S}$ accepts the supremal controllable language of $R_f(\Sigma)$. The objective in supervisory control synthesis is to find the admissible maximally permissive supervisor $\mathcal{S}$.

Consider a net system Σ with Petri net $\mathcal{N} = (S, T, F)$. A *monitor* supervised system is an augmented net system Σ_m in which $\mathcal{N} = (S \cup S_m, T, F \cup F_m)$. The additional places S_m are called monitors which are connected to the rest of the net system transitions in T through the additional arcs in F_m. Monitors act as semaphores or mutexes that can help enforce generalized mutual exclusion constraints on the original net system [3]. Monitored net systems represent a special class of Petri-net based supervised system which have been well studied in recent years [3, 13, 14, 8]. A large portion of their popularity rests with the fact [13] that monitors enforcing inequality constraints on the marking vector can be readily determined using a simple matrix computation. For net systems in which all transitions are controllable, the resulting monitor is maximally permissive. If the controllable transitions are a proper subset of T, however, the resulting monitor is not necessarily maximally permissive [14, 8]. The principle results of this paper allow us to transform maximally permissive liveness-enforcing marking based supervisors into monitors and we identify specific conditions under which the resulting monitor supervisor is not maximally permissive.

3 Liveness Enforcing Supervisors

This section summarizes basic results [6, 7] concerning unfolding and its application to Petri net supervision.

3.1 Network Unfolding

Unfolding is a specific type of partial order method that is useful in alleviating the state-explosion problems encountered in verifying system properties. An unfolding is a map (net homomorphism) from a special acyclic finitary Petri net known as an occurrence net back onto the original Petri net. The map is defined in such a way that the causal relationships between transitions is preserved, thereby allowing us to construct a reduced reachability graph that enumerates all reachable marking of the original net system. In many cases, this reduced reachability graph (i.e. the occurrence net) can greatly reduce the complexity of the verifi-

cation problem [12]. The potential value of unfolding has long been recognized [2, 9, 6] as have some other related partial order methods [5].

Specifically, an unfolding unfolds the original net system into an acyclic occurrence net starting from the initial marking. A net homomorphism h was developed to map places and transitions in the occurrence net to their original versions in the original net system. h also preserves the presets and postsets of transitions in the original net system.

The unfolding may be infinite, since the original net system may be cyclic. It thus make sense to develop a *finite prefix* to truncate the unfolding. Let the *cause* of a transition t, denoted as $[t]$, in the occurrence net be the set of all transitions that either precede or equal t. The *cut* of $[t]$, denoted as $Cut([t])$, is a set of places in the occurrence net that can be mapped to the marking reached after firing (the image of) all transitions in $[t]$ in the original net system. A transition t is called a *cut-off transition* if there is a smaller cause with the same cut as that of $[t]$. A finite prefix of the unfolding is formed by removing all transitions that succeed a cut-off transition. It was proven in [12] that the finite prefix enumerates all the reachable markings of the original net system.

Let $\beta_c = (\mathcal{N}_c, h_c)$ be the finite prefix of the unfolding of $(\mathcal{N}, \mu_0)$. In $\mathcal{N}_c = (S_c, T_c, F_c)$, we define the set of *end transitions* as $T^{end} = \{t \in T_c \mid \ \nexists t' \in T_c, \text{ such that } t < t'\}$. Let T^{cut} denote the set of cut-off transitions in T_c. It is obvious that $T^{cut} \subseteq T^{end}$. For any $t^e \in T^{end}$, define a *base configuration* BC^{t^e} as the cause of t^e, i.e. $BC^{t^e} = [t^e]$.

As an example, consider the net system in figure 1. The right-hand net is the occurrence net associated with the prefix β_c and the left-hand net is the original net system $(\mathcal{N}, \mu_0)$. In particular, there are three end transitions $\{t'_3, t'_4, t'_5\}$. Note that the cut of configuration $[t'_4]$ and $[t'_5]$ enables no transition in the original net system. The cut of configuration $[t'_3]$, however, remarks the initial marking μ_0 of $\mathcal{N}$ and forms a cycle.

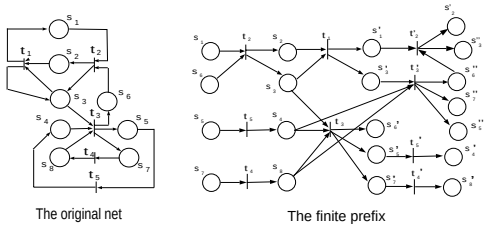

Figure 1: An example net and its unfolding

3.2 Liveness and Supervision

Since a suitably defined prefix of an unfolding enumerates all reachable markings in the net system, the prefix can be used to characterize properties of the net system. In [6], unfoldings were used to verify whether or not the net system was live. In [7], such unfoldings were also used to construct liveness enforcing marking based supervisors for ordinary Petri nets. This section reviews these prior results.

The principle result of [6] proves that the liveness of a bounded ordinary Petri net can be assessed by investigating the sequential and concurrent executions of base configurations. The sequential execution of a set of base configurations is said to be live, if either it fires every transition in the original system, or it repeatedly reaches the initial marking. A base configuration is said to be live (deadlocked), if it is contained (not contained) in a live sequential execution of base configurations. The concurrent execution of a set of base configurations is said to be live, if it does not contain a sequence of transitions that causes a *cyclic lock* where no transition in any of these base configurations is enabled. It was proven in [6] that a bounded Petri net is live if and only if every base configurations of $\mathcal{N}_c$ is live and there does not exist a set of base configurations in cyclic lock.

Deadlocked and cyclic-locked base configurations can be viewed as local deadlocks in the net system. This information can be used to develop maximally permissive marking based supervisory policies that enforce the net system's liveness. The principle result in [7] proves that the liveness of a bounded Petri net can be enforced by control-disabling every *critical transition* t that leads to a local deadlock at every *critical marking* that enables t. Specifically, a transition t in the occurrence net $\mathcal{N}_c$ is called a *critical transition*, if $[t]$ is not a subset of any live base configuration but $[t] - t$ is a subset of some live base configuration. A transition t is called a *lock set transition* if t is contained in a sequence of transitions causing cyclic lock. A marking μ is called a *critical marking*, if it enables either a critical transition or a lock set transition. It is shown in [7] that the maximally permissive supervisor can be constructed by disabling critical or lock set transitions at their critical markings.

A critical transition t may not be controllable, but it is still possible to control-disable it by control disabling certain transitions t' in its cause. There may, of course, be multiple controllable transitions in $[t]$ to choose from. One rule for choosing t' is that it should not disable any other live base configuration. We say a critical or lock set transition t is *cause controllable* if there exists such a $t' \in [t]$. We say a deadlocked base configuration is controllable if its critical transition is cause controllable. A cyclic lock is said to be controllable if every lock set transition is cause controllable. It was proven in [7] that for an n-safe net system $(\mathcal{N}, \mu_0)$, there exists a maximally permissive supervisory pol-

icy that enforces liveness if and only if every deadlock base configuration is controllable and every cyclic lock is controllable.

4 Main Result

This section provides two procedures that transform maximally permissive liveness-enforcing marking based supervisors to liveness-enforcing monitors. The first procedure assumes that there only exist deadlocked base configurations in the occurrence net and the second one assumes that there only exist cyclic locks. Combining the two procedures, we prove that a maximally permissive liveness-enforcing marking based supervisor can always be transformed to a liveness-enforcing monitor. We also provide examples to illustrate some cases where the liveness-enforcing monitor is not maximally permissive.

4.1 Disabling Critical Transitions using Monitors

Let t_r be a critical transition in the occurrence net. Let BC^l be a live base configuration such that some transition in BC^l is in conflict with t_r. A transition t_r^o is called a *choice transition* if $t_r^o \# t_r$ and $\nexists t < t_r^o, t \# t_r$. In other words, t_r^o is the "front-most" transition in BC^l that is in conflict with t_r. The exact procedure that disables critical transitions using monitors is presented below. The intuition behind this procedure is to introduce a mutual exclusion between t_r and a transition $t^p \in BC^l$ that precedes both t_r and t_r^o.

Procedure 1. Disable critical transitions using monitors

1. In the occurrence net $\mathcal{N}_c$, find all deadlocked base configurations and do the following.
2. Pick one uncontrolled deadlocked base configuration BC^d and find its critical transition t_r.
3. Find a live base configuration BC^l that contains a choice transition t_r^o.
4. In the original net, find a transition t^p such that $t^p \in \bullet(\bullet h_c(t_r^o) \cap \bullet h_c(t_r))$. Pick a controllable transition $t^d \in T^{[t_r]}$ such that $\nexists t' \in T^{[t_r]}, t^d < t'$. If $t \neq t_r$ (t_r is uncontrollable), then in the original net, add a control place s_c' and arcs $(s_c', h_c(t^d))$, (t^p, s_c'). In the original net, add a control place s_c and control arcs (s_c, t^p), $(s_c, h_c(t^d))$, $(h_c(t_r^o), s_c)$ and $(h_c(t^d), s_c)$. If in $\mathcal{N}_c$, there is a transition $t_c^p \in \bullet(\bullet t_r^o \cap \bullet t_r))$, such that $h_c(t_c^p) = t^p$, then initially s_c contains one token and s_c' (if exists) is empty, otherwise s_c is initially empty and s_c' contains one token.
5. if there exists another choice transition t_r^o for t_r, goto step 3.
6. if there exist other uncontrolled deadlocked base configurations, goto step 2.

Following procedure 1, the liveness-enforcing monitors for the example net in figure 1 are shown in figure 2 and figure 3. Figure 2 shows the maximally permissive monitor when the critical transition t_3 is controllable. Figure 3 shows two possible liveness-enforcing monitors when t_3 is uncontrollable. It is easy to see that neither of these two possible monitors is maximally permissive.

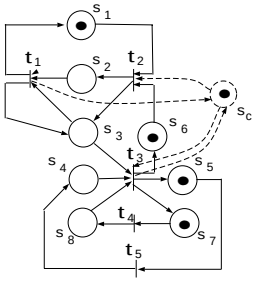

Figure 2: The deadlock-controlled net

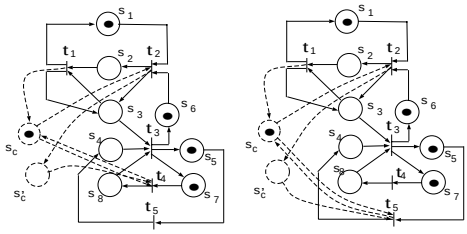

Figure 3: The two possible monitors

Some properties of the monitor constructed using procedure 1 are proven below.

Lemma 4.1 *The monitor constructed using procedure 1 eliminates all the deadlocked base configurations and enforces all the live base configurations.*

Proof: The prove is straightforward. For any deadlocked base configuration, first consider the case where the critical transition $h_c(t_r)$ is controllable. We only need to prove every critical transition $h_c(t_r)$ is control-disabled at every critical marking. First observe that $h_c(t_r)$ cannot fire when t^p is enabled, since $t^p \in \bullet(\bullet h_c(t_r))$. Secondly, $h_c(t_r)$ still cannot fire right after the firing of t^p, since $s_c \in \bullet h_c(t_r)$ and s_c is empty after t^p fires. Note also that the consecutive firing of $t^p h_c(t_r^o)$ is not affected by the monitor. Based on these observations we concludes that whenever there is a choice between $h_c(t_r)$ and $h_c(t_r^o)$, $h_c(t_r)$ is always control-disabled by the monitor and this proves that

when the critical transition is controllable, the deadlocked configuration is removed by the monitor.

Next, consider the case when the critical transition is uncontrollable. Note that the addition of control place s'_c and arcs $(s'_c, h_c(t^d))$, (t^p, s'_c) introduce a choice between t^o_r and t^d. Following the proves above, we know that the addition of control place s_c and control arcs (s_c, t^p), $(s_c, h_c(t^d))$, $(h_c(t^o_r), s_c)$ and $(h_c(t^d), s_c)$ disables $h_c(t^d)$ whenever there is a choice between $h_c(t^o_r)$ and $h_c(t^d)$. Since $t_d < t_r$, then $h_c(t_r)$ is also control-disabled whenever $h_c(t^d)$ is control-disabled. This means that the deadlocked base configuration ($h_c(t_r)$ uncontrollable) is removed by the monitor. •

Lemma 4.2 *If there does not exist cyclic lock in the original net system, then there does not exist any liveness-enforcing monitor supervisor that is more permissive than the one constructed using Procedure 1.*

Proof: The prove is also straightforward. By lemma 4.1, if $h_c(t_r)$ is controllable, then the monitor constructed using procedure 1 control-disables $h_c(t_r)$ at its critical marking. This means that the monitor is maximally permissive when $h_c(t_r)$ is controllable. If $h_c(t_r)$ is uncontrollable, then the monitor control-disables $h_c(t^d)$, which is an "immediate" controllable predecessor of $h_c(t_r)$. Since there is no transition $t \in T^{[t_r]}$ in the occurrence net such that $t^d < t < t_r$, then there does not exist any language-preserving monitor that is more permissive than the one constructed using procedure 1. •

Lemma 4.3 *If there does not exist cyclic lock in the original net system, then the monitor constructed using procedure 1 is maximally permissive if and only if for any uncontrollable critical transition $h_c(t_r)$, there exists only one transition $t^d \in T^{[t_r]}$ in the occurrence net such that $\nexists t \in T^{[t]}, t^d < t$.*

Proof: Sufficiency: If $h_c(t_r)$ is uncontrollable and there is only one such t^d in $T^{[t_r]}$, then in the controlled net, $h_c(t^d)$ is control-disabled by s_c until $t^p, h_c(t^o_r)$ are fired. If there exists a more permissive supervisor that control-enables $h_c(t^d)$ before $h_c(t^o_r)$ is fired, then $t^p h_c(t_r)$ may fire consecutively (since once $h_c(t^d)$ is fired $h_c(t_r)$ cannot be control-disabled) and therefore the original net system will have a deadlock.

Necessity: If there exist two such t^d's, denoted as t^d_1, t^d_2, then we can chose either one of them to control. Since we have this choice, then the monitor constructed using either t^d_1 or t^d_2 is not maximally permissive. •

4.2 Disabling Cyclic Locks using Control Places

This subsection presents a procedure that produces control places that disable cyclic locks. The idea in this procedure is to use control places to introduce mutual exclusion between lock set transitions so that they cannot fire consecutively. The exact procedure is presented below.

Procedure 2. Disable cyclic locks using monitors

1. In the occurrence net $\mathcal{N}_c$, find all combinations of live base configurations that are in cyclic lock and do the following until the procedure exhausts them.
2. Pick one cyclic locked set of live base configurations and find its lock set $t^c_1, \cdots, t^c_k$ and also $t^{c\prime}_1, \cdots, t^{c\prime}_k$.
3. In the original net, add a control place s_L, pick two lock set transitions t^c_i, t^c_j, if $t^c_i(t^c_j)$ is controllable, add arc $(s_L, h_c(t^c_i))$ $((s_L, h_c(t^c_j)))$ and arc $(h_c(t^{c\prime}_i), s_c)$ $((h_c(t^{c\prime}_i), s_c))$. If $t^c_i(t^c_j)$ is uncontrollable, then find a transition $t^d_i(t^d_j) \in T^{[t^c_i]}(T^{[t^c_j]})$ such that $\nexists t \in T^{[t^c_i]}(T^{[t^c_j]})$ such that $t^c_i(t^c_j) < t$. Add arcs $(s_L, h_c(t^d_i))$ $((s_L, h_c(t^d_j)))$ and $(h_c(t^{c\prime}_i), s_c)$ $((h_c(t^{c\prime}_i), s_c))$.
4. The initial marking in control place s_L is 1.

As an example, consider the net in figure 4. This figure shows a net containing three live base configurations $BC^{t_4}, BC^{t_5}, BC^{t_6}$ in cyclic lock. The lock set transitions are t_1, t_2, t_3. In this case we have three pairs of lock set transitions to chose from to enforce mutual exclusion. The dotted part in figure 4 shows the monitor that enforce the mutual exclusion between t_1, t_2. This monitor supervisor is less permissive than the maximally permissive marking based supervisor, since the marking based supervisor permits the marking after the firing of t_1, t_2, (marking $[010110]^T$), while the monitor does not permit it. In fact, the set of markings reached under the maximally permissive marking based supervisor is the union of the markings reached under any one of the three monitors.

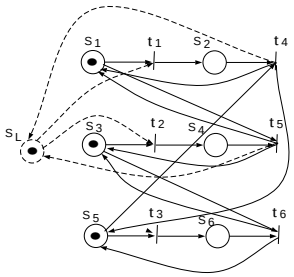

Figure 4: The tri-cyclic lock net

Lemma 4.4 *The monitor constructed using procedure 2 eliminates all the cyclic locks and enforces all the live base configurations.*

Proof: Note that in the controlled net, what s_L does is to introduce a mutual exclusion between t_i^c and t_j^c. If t_i^c is fired, t_j^c can not fire until $t_i^{c\prime}$ is fired since s_L is not marked again until $t_i^{c\prime}$ is fired. This means we cannot fire all the lock set transitions consecutively and this avoids the occurrence of the cyclic lock. •

The following two theorems summarize the main result.

Theorem 4.1 *For an n-safe ordinary Petri net, there exists a liveness-enforcing monitor if and only if there exists a maximally permissive marking-based liveness-enforcing supervisory.*

Proof: Sufficiency: Directly follows from lemma 4.1 and lemma 4.4, since all the control places introduced only disable deadlocks and cyclic locks and do not affect the live base configurations.

Necessity: Trivial, since a maximally permissive marking based liveness-enforcing supervisor includes any liveness-enforcing monitor. •

Theorem 4.2 *For an n-safe ordinary Petri net, a live-enforcing monitor may not be maximally permissive.*

Proof: Directly follows from lemma 4.3 and the examples in figure 3 and figure 4. •

5 Conclusion

This paper discusses the transformation of maximally permissive liveness-enforcing marking based supervisors to monitors. Determining necessary and sufficient conditions for the existence of maximally permissive monitors that disable cyclic locks is among the topics of future research work.

References

[1] J. Desel and J. Esparza, Free Choice Petri Nets, *Cambridge Tracts in Theoretical Computer Science 40*, Cambridge University Press 1995.

[2] Esparza, J., S. Romer and W. Vogler . An improvement of McMillan's unfolding algorithm. In: vol. 1055 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[3] Giua, A., Dicesare, F., , and Silva, M., "Generalized mutual exclusion constraints on nets with uncontrollable transitions", *Proceedings of the 1992 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 974-979, Chicago, IL.

[4] Blocking and Controllability of Petri Nets in Supervisory Control, IEEE Trans. on Automatic Control, Vol 39(4), 1994.

[5] P. Godefroid, Partial Order Methods for the Verification of Concurrent Systems, Ph.D. Thesis, 1994, University de Liege.

[6] K.X. He and M.D. Lemmon, "Liveness verification of discrete-event systems modeled by n-safe Petri nets". In the *Proceedings of the 21st International Conference on Application and Theory of Petri Nets*, Danmark, June, 2000.

[7] K.X. He and M.D. Lemmon, "On the existence of liveness-enforcing supervisory policies of discrete-event systems modeled by n-safe Petri nets". In the *Proceedings of 2000's IFAC conference on Control Systems Design, special session on Petri nets*, Slovakia, June, 2000.

[8] M.V. Irodache, "A method for deadlock prevention in Discrete Event Systems using Petri nets", Technical Report of the ISIS Group at the University of Notre Dame, ISIS-99-006, July, 1999.

[9] A. Kondratyev, M. Kishinevsky, A. Taubin and S. Ten, "Structural approach for the analysis of Petri nets by reduced unfoldings", *Proceedings of the 17th International Conference on Application and Theory of Petri Nets*, Osaka, Japan, June 24-28, 1996.

[10] Li, Y., and Wonham, W. M., "Control of vector discrete-event systems I-the base model", *IEEE Transactions on Automatic Control*, vol. 38, no. 8, pp. 1214-1227, 1993. Correction in *IEEE Transactions on Automatic Control*, vol. 39, no. 8, pp.1771, Aug, 1994.

[11] Li, Y., and Wonham, W. M., "Control of vector discrete-event systems II-controller synthesis", *IEEE Transactions on Automatic Control*, vol. 39, no. 3, pp. 512-530, 1994.

[12] McMillan, K., *Symbolic Model Checking*, Kluwer Academic Publishers, 1993.

[13] Moody, J.O., Yamalidou, K., Lemmon, M.D. and Antsaklis, P.J., "Feedback control of Petri nets based on place invariants", *Proceedings of the 33rd IEEE CDC*, vol 3, pp. 3104-3109, Lake Buena Vista, FL.

[14] Moody, J.O. and Antsaklis, P.J., "Supervisory control using computationally efficient linear techniques: A tutorial introduction", In *Proceedings of the 5th IEEE Mediterranean Conference on Control and Systems*, Paphos, Cyprus, July, 1997.

[15] Murata, T., "Petri nets: Properties, analysis, and applications", *Proceedings of the IEEE*, 77(4):541-580.

[16] Reisig, W. (1985). *Petri Nets*. Springer-Verlag.

[17] R.S. Sreenivas, On the existence of supervisory control in discrete event dynamic systems modeled by controlled Petri nets, *IEEE Trans. on Automatic Control, 42(7), July, 1997, pp. 928-945.*