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

Marian V. Iordache
Department of
Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556
iordache.1@nd.edu

John O. Moody
Lockheed Martin
Federal Systems
1801 State Rt. 17C, MD 0210
Owego, NY 13827-3998
john.moody@lmco.com

Panos J. Antsaklis
Department of
Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556
antsaklis.1@nd.edu

Interdisciplinary Studies of Intelligent Systems

A METHOD FOR DEADLOCK PREVENTION IN DISCRETE EVENT SYSTEMS USING PETRI NETS

Marian V. Iordache*, John O. Moody†, Panos J. Antsaklis*

Abstract

Deadlock is the condition of a system that has reached a state in which all of its potential actions are blocked. This paper introduces a deadlock prevention method for discrete events systems modeled by Petri nets. Petri nets have a bipartite graph structure and they are particularly well suited to model concurrencies found in manufacturing, communication and computer systems, among others. Given an arbitrary Petri net structure, the deadlock prevention algorithm in this paper finds linear inequalities in terms of the marking (state vector). When the Petri net is supervised according to the constraints provided by the algorithm, the supervised net is proved to be deadlock-free for all initial markings that satisfy the supervision constraints. Results pertaining to permissivity properties and termination are also proved. The algorithm is applicable to any Petri net with controllable and observable transitions.

1 Introduction

Deadlock is the state of a system in which no action can take place. This paper intorduces an algorithm for the prevention of (global) deadlock in systems modeled with Petri nets. The algorithm is not meant to enforce liveness, i.e. to prevent local deadlock, in which only a part of the system is deadlocked, although in some cases it might enforce liveness as well. Liveness enforcement is a stronger requirement than deadlock prevention, because a system might not be in deadlock when some of its subsystems are deadlocked.

Deadlock usually appears in systems that contain subsystems that run in parallel and share some form of common resources. Because Petri nets are a formal model of concurrent systems, they are appropriate for deadlock study.

The deadlock prevention method of this work is described by an iterative algorithm, whose purpose is to find linear inequalities in terms of the marking vector, the state variable of Petri nets, such that whenever these inequalities are satisfied, the Petri net is not in deadlock. A Petri net

*Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556 (e-mail: iordache.1, antsaklis.1@nd.edu)

†Lockheed Martin Federal Systems, 1801 State Rt.17C, MD 0210, Owego NY 13827-3998 (e-mail: john.moody@lmco.com)

supervisor that enforces these linear inequalities is built using an established technique. Some of the results are concerned with the permissivity of the supervisor and a sufficient condition for the algorithm to terminate. The most important result guarantees (under appropriate conditions) that the Petri net supervised according to the linear constraints found by the algorithm is deadlock-free for all initial markings that satisfy these constraints.

Differences from other deadlock prevention approaches are that the initial marking is not assumed to be known, but rather it is regarded as a parameter. There is a related approach that does not explicitly require the initial marking to be known [Lautenbach, 1996], but it works for less general Petri net structures. The only requirement we make on the Petri net structure is that all transitions are controllable and observable, because the other case was not yet investigated. Research to generalize the method to still more general Petri nets is in progress.

Researchers have used varied system models depending on the applications they were studying. Some other models that are not obviously related to Petri nets are the models of *finite automaton* type and the *resource allocation graph*.

A resource allocation graph [Sinha, p.308] is a bipartite graph used to define the state of a set of processes with common resources. Applications include Operating Systems, in Computer Science.

Finite automata are the models most used in the design of discrete event systems. Their simplicity allows solving many design problems. However, they are sequential models, and the number of states in real applications may become too large for computations to be done in a reasonable amount of time.

Petri nets have a bipartite graph structure. Unlike resource allocation graphs, their structure is fixed. The change of the Petri net state is described by the *marking vector*. The Petri net is a more powerful model than the finite automaton.

There are various methods proposed for deadlock prevention or deadlock avoidance in the literature. Deadlock avoidance requires little or no off-line computation, relying mostly on on-line computations. An algorithm is used to check in real time what actions could be performed. Deadlock prevention on the other hand relies on off-line computation and performs almost no on-line computation. Deadlock prevention is a true real-time solution, but some researchers regard deadlock avoidance as less restrictive.

Deadlock avoidance methods are in many cases related to the resource allocation algorithm of [Dijkstra, 1965] and/or the necessary conditions formulated in [Coffman, 1971]. In [Banszak] deadlock avoidance is considered in manufacturing systems modeled using a particular form of ordinary Petri nets. [Fanti] uses digraph models; note that digraphs are less general than Petri nets. [Reveliotis] considers polynomial complexity policies for sequential resource allocations systems, where the most general model, not considered there, can be modeled with Petri nets. The method from [Lewis] considers manufacturing systems which can be modeled with Petri nets and does not guarantee deadlock-freedom when a type of cyclic structure is included. [Barkaoui, 1995] considers conservative Petri net models and does not guarantee deadlock-freedom.

Deadlock prevention methods typically use structural properties of the net. Deadlock in Petri

nets was related to *siphons*, a set of *places* with a specific property (section 2). Liveness was also found to depend on siphons for the class of *free-choice Petri nets* in [Hack, 1972] and [Commoner, 1972]. More recently a similar relation was found for the more general *asymmetric-choice Petri nets* in [Barkaoui, 1996]. Algorithms for siphon computations can be found in [Lautenbach, 1987] and [Ezpeleta, 1993]. Among deadlock prevention papers we mention [Lautenbach, 1996] and [Ezpeleta, 1995].

Papers which explicitly use control places to restrict the behavior of a Petri net include [Suraj], [Giua, 1992], [Moody, 1994] and [Yamalidou, 1994]. Control places were also used for siphon control in [Barkaoui, 1995], [Ezpeleta, 1995] and [Lautenbach, 1996]. In this paper the supervisors are built using the place invariant methodology from [Yamalidou] and [Moody, 1998].

[Ezpeleta, 1995] also addresses deadlock prevention in flexible manufacturing systems. The Petri net model that is used is called S^3PR, which is ordinary and conservative. The paper finds a control policy which uses control places to enforce liveness. The advantages of the method are simplicity and the guarantee success. A disadvantage is that the supervision is relatively restrictive.

[Lautenbach, 1996] is the paper most related to our approach. It has a similar iterative process in which every new minimal siphon is controlled. Its unique feature among other deadlock papers is that it works with Petri nets that are not assumed to be ordinary. A transformation to almost ordinary Petri nets is used. We also use it in a slightly simplified form (section 4.3). The problem of source places, which is likely to appear for Petri nets that are not repetitive, is not considered in [Lautenbach, 1996].

The method for deadlock prevention introduced in this paper has the considerable advantage of having guaranteed performance and being applicable to any Petri net. This is in contrast to previous approaches, that are either applicable to restricted classes of Petri nets and/or without proofs. In the approach presented in this paper, when the algorithm terminates, deadlock prevention is guaranteed under certain conditions. In the common in practice case of structurally bounded Petri nets, our deadlock prevention algorithm is shown to terminate under certain conditions. The algorithm generates a set of linear inequality constraints which when implemented via supervision, guarantee deadlock prevention. In special cases, the supervisor will also enforce liveness, in which case the liveness enforcing supervisor is maximally permissive.

We begin with a short review of Petri net definitions and properties in section 2. Section 3 presents deadlock properties and in section 3.2 some original deadlock enforcement results are given. Related work appears in [Sreenivas, 1997], a paper concerned with existence of supervisory policies for liveness enforcement. Section 4.1 outlines the method of enforcing linear constraints from [Moody, 1998]. The deadlock prevention algorithm is formulated in section 4 and performance results are given in section 5.

2 Review of Some Petri Net Basic Properties

In this paper we assume that the reader knows the fundamentals of Petri nets. Good introductions to Petri nets are for instance [Murata], [David] and [Reisig]. This section is meant mainly to

introduce our notations.

A **Petri net structure** is a quadruple $\mathcal{N} = (P, T, F, W)$ where P is the **set of places**, T the **set of transitions**, $F \subseteq (P \times T) \cup (T \times P)$ is the set of **transition arcs** and $W : F \to \mathbb{N} \setminus \{0\}$ is a **weight function**. A **marking** μ of the Petri net structure is a map $\mu : P \to \mathbb{N}$. A Petri net structure $\mathcal{N}$ with **initial marking** μ_0 is called a **Petri net**, and will be denoted by $(\mathcal{N}, \mu_0)$. For simplicity, we may denote sometimes by Petri net a Petri net structure.

It is useful to consider a marking both as a map and as a vector. These requirements are not necessarily conflicting, because there are authors ([Reisig]) that define vectors as maps defined on a set A instead of $\{1, 2, \dots m\}$, as is customary. The **marking vector** is defined to be $[\mu(p_1), \mu(p_2), \dots \mu(p_n)]^T$, where $p_1, p_2, \dots p_n$ are the places of the net enumerated in a chosen (but fixed) order and μ the current marking. The same symbol μ will denote a marking vector. The marking vector of a Petri net may be regarded as the state variable of the Petri net. An equivalent way of saying that place p has the marking $\mu(p)$ is that p has $\mu(p)$ **tokens**.

Figure 1 could be used to illustrate the graphical representation of Petri nets. A token is represented by a bullet. The marking vector in figure 1(b) is $[0, 1, 1]^T$. An arc weight is indicated near the arc when it is not one. For instance, in figure 1(b) $W(p_3, t_1) = 2$ and $W(t_2, p_2) = 4$.

The **preset** of a place p is the set of incoming transitions to p: $\bullet p = \{t \in T : (t, p) \in F\}$. The **postset** of a place p is the set of outcoming transitions from p: $p\bullet = \{t \in T : (p, t) \in F\}$. p is a **source place** if $\bullet p = \emptyset$ and a **sink place** if $p\bullet = \emptyset$. Similar definitions apply for transitions. They are also extended for sets of places or transitions; for instance, if $A \subseteq P$, $\bullet A = \bigcup_{p \in A} \bullet p$, $A\bullet = \bigcup_{p \in A} p\bullet$.

We use $\mu[t$ to denote that μ enables the transition t and $\mu[t > \mu'$ to denote that μ enables t and if t fires, then the marking becomes μ'. The marking μ' is **reachable** from μ if there is a sequence of markings $\mu_1, \dots \mu_k$, $\mu_k = \mu'$, and a sequence of transitions $t_{i_1}, \dots t_{i_k}$ s.t. $\mu[t_{i_1} > \mu_1[\dots t_{i_k} > \mu'$. The **set of reachable markings** of a Petri net $(\mathcal{N}, \mu)$ (i.e. the set of markings reachable from the initial marking μ) will be denoted by $\mathcal{R}(\mathcal{N}, \mu)$.

In a Petri net $\mathcal{N} = (P, T, F, W)$ with m places and n transitions, the **incidence matrix** is an $m \times n$ matrix defined by $D = D^+ - D^-$, where the elements d_{ij}^+ and d_{ij}^- of D^+ and D^- are

$d_{ij}^+ = W(t_j, p_i)$ if $(t_j, p_i) \in F$ and $d_{ij}^+ = 0$ otherwise;

$d_{ij}^- = W(p_i, t_j)$ if $(p_i, t_j) \in F$ and $d_{ij}^- = 0$ otherwise.

The incidence matrix allows an algebraic description of the marking change of a Petri net:

$$\mu_k = \mu_{k-1} + D \cdot u_k \tag{1}$$

where u_k is called **firing vector**, and its elements are all zero excepting $u_{k,i} = 1$, where i corresponds to the transition t_i that fired. We will denote by **firing vector** also a vector x associated with a sequence of transitions that have fired, whose entries record how often each transition appears in the sequence. If x is the firing vector of the transition sequence that led the Petri net from the marking vector μ_0 to μ_k:

$$\mu_k = \mu_0 + D \cdot x \tag{2}$$

A vector x is called **place invariant** if $x^T \cdot D = 0$. A vector x is called **transition invariant** if $D \cdot x = 0$. The **support of a transition invariant** x is $||x|| = \{t_j \in T : x(j) \neq 0\}$.

A Petri net $(\mathcal{N}, \mu_0)$ is said to be **deadlock-free** if for any reachable marking μ there is an enabled transition. $(\mathcal{N}, \mu)$ is in **deadlock** if no transition is enabled at marking μ.

Let $(\mathcal{N}, \mu_0)$ be a Petri net. A transition t is said to be **live** if $\forall \mu \in \mathcal{R}(\mathcal{N}, \mu_0)\ \exists \mu' \in \mathcal{R}(\mathcal{N}, \mu)$ such that t is enabled by μ'. A transition t is **dead** at marking μ if no marking $\mu' \in \mathcal{R}(\mathcal{N}, \mu)$ enables t. $(\mathcal{N}, \mu_0)$ is said to be **live** if every transition is live.

A nonempty set of places $S \subseteq P$ is called a **siphon** if $\bullet S \subseteq S\bullet$ and **trap** if $S\bullet \subseteq \bullet S$. An **empty siphon** with respect to a Petri net marking μ is a siphon S such that $\sum_{p \in S} \mu(p) = 0$. The attribute "empty" refers to the fact that S has no tokens. A siphon has the property that if for some marking it is empty, it will be so for all subsequent reachable markings. A trap has the property that if at some marking it has one token, then for all subsequent reachable markings it will have at least one token. See figure 1 for siphon examples. In figure 1(a), $\{p_1, p_3\}$ and $\{p_2, p_4\}$ are traps. S is a **minimal siphon** if there is no other siphon S' (by definition, $S' \neq \emptyset$) such that $S' \subset S$.

3 Deadlock and Liveness Properties of Petri Nets

This section introduces certain liveness and deadlock properties, focusing on their relation to structural properties of Petri nets and supervision. All transitions are considered to be controllable and observable.

3.1 Intrinsic Properties

A Petri net $\mathcal{N} = (P, T, F, W)$ is **ordinary** if $\forall f \in F :\ W(f) = 1$. In the specification of our results we will refer to slightly more general Petri nets in which only the arcs from places to transitions have weights equal to one. We are going to call such Petri nets *PT-ordinary*, because all arcs (p, t) from a place p to a transition t satisfy the requirement of an ordinary Petri net that $W(p, t) = 1$.

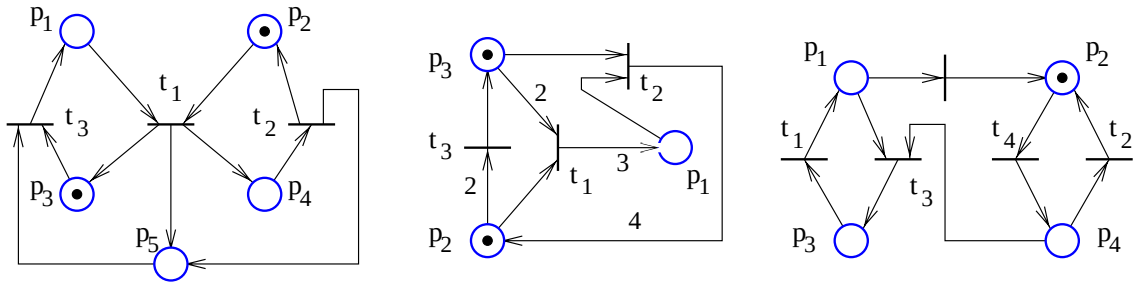
Definition 3.1 *Let $\mathcal{N} = (P, T, F, W)$ be a Petri net. We call $\mathcal{N}$* **PT-ordinary** *if $\forall p \in P, \forall t \in T$, if $(p, t) \in F$ then $W(p, t) = 1$.*

The basis of the results of this paper comes from a well known necessary condition for deadlock ([Reisig]), namely that a deadlocked ordinary Petri net contains at least one empty siphon. It can easily be seen that the proof of this result also is valid for PT-ordinary Petri nets and so the following proposition follows:

Proposition 3.1 *A deadlocked PT-ordinary Petri net contains at least one empty siphon.*

An example is shown in figure 1(a). A simple way to generalize this result to more general Petri nets is given in Proposition 3.2. The proof of Propositions 3.1 and 3.2 are similar.

Proposition 3.1 shows that deadlock might be prevented if it can be ensured in a nonblocking way that no siphon ever loses all its tokens. The condition in Proposition 3.1 is only necessary. The example of figure 1(c) illustrates that the condition of Proposition 3.1 is not sufficient and figure 1(b) that the result is not applicable to Petri nets more general than PT-ordinary.



(a) (b) (c)

Figure 1: (a) A deadlocked PT-ordinary Petri net. An empty siphon is $\{p_1, p_4, p_5\}$. (b) A deadlocked Petri net with no empty siphon which is not PT-ordinary. (c) A deadlock-free Petri net (for the marking displayed) with an empty siphon – $\{p_1, p_3\}$.

Definition 3.2 (cf. [Barkaoui]) *Let $\mathcal{N}$ be a Petri net and M a marking. $\mathcal{N}$ is said to be* **well-marked** *for M if in every siphon there is at least a token.*

Definition 3.3 *Let $\mathcal{N}$ be a Petri net and $\mathcal{M}_I$ be a set of initial markings. A siphon S is said to be* **controlled** *with respect to $\mathcal{M}_I$ if $\forall \mu_0 \in \mathcal{M}_I$, $\forall \mu \in \mathcal{R}(\mathcal{N}, \mu_0)$: $\sum_{p \in S} \mu(p) \geq 1$.*

A controlled siphon contains for all reachable markings at least one token. A **trap controlled siphon** is a siphon that includes a trap. Recalling the trap property, for all markings such that the trap has one token, the siphon is controlled.

We define an **invariant controlled siphon** as a siphon S of a Petri net $\mathcal{N}$ with the property that $\mathcal{N}$ has a place invariant x such that for all $i = 1, 2, \ldots |P|$, if $x(i) > 0$ then $p_i \in S$. It is easy to show that for all initial markings μ_0, such that $x^T \mu_0 \geq 1$, the siphon S is controlled.

In particular, a siphon which contains a controlled siphon is controlled. Therefore in a Petri net such that all minimal siphons are controlled, all siphons are controlled. Also, by Proposition 3.1, a PT-ordinary Petri net is deadlock-free if all its siphons are controlled. This is not true for more general Petri nets. The following result is also in [Barkaoui, 1996].

Proposition 3.2 *A deadlocked Petri net with marking μ has at least one siphon S such that $\forall p \in S$ $\exists t \in p\bullet$ with $W(p,t) > \mu(p)$.*

Proof: Let S be the set of all places p such that $\exists t \in p\bullet : \mu(p) < W(p,t)$. Then, $\forall t \in T$, $t \in S\bullet \Rightarrow T \subseteq S\bullet$. Obviously, $\bullet S \subseteq T \Rightarrow \bullet S \subseteq S\bullet$, and so S is a desired siphon. □

Figure 1(b) shows a deadlocked Petri net. There are two minimal siphons: $S_1 = \{p_1, p_2\}$ and $S_2 = \{p_2, p_3\}$. The marking of p_3 does not prevent t_2 from firing but does prevent t_1. The marking of p_2 does not prevent t_1 but prevents t_3. For the current marking $[0,1,1]$, both siphons S_1 and S_2 satisfy the necessary condition of the proposition. For the deadlock the marking $[0,0,2]$, only one of them satisfies it. The requirement of Proposition 3.2 seems difficult to relax. For instance, it is not true that if in all minimal siphons S, if $\exists p \in S\ \forall t \in p\bullet \cap \bullet S$, $\mu(p) \geq W(p,t)$ then the Petri net is not in deadlock, as it could be checked in figure 1(b).

Loss of liveness is a less severe form of deadlock, where some actions can no longer happen while others may still be possible. Deadlock implies loss of liveness. An empty siphon is a necessary and not a sufficient condition for deadlock, while for loss of liveness it is a sufficient but not a necessary condition. Commoner's Theorem states that in an ordinary free choice net $\mathcal{N}$, if there are dead transitions for a marking μ, then there is a reachable marking $\mu' \in \mathcal{R}(\mathcal{N}, \mu)$ such that a siphon is empty ([Reisig, p.103]). Theorem 3.1 is the generalization to asymmetric choice nets. An **asymmetric choice** net is a Petri net $\mathcal{N} = (P, T, F, W)$ with the property that $\forall p_1, p_2 \in P$, $p_1\bullet \cap p_2\bullet \neq \emptyset \Rightarrow p_1\bullet \subseteq p_2\bullet$ or $p_2\bullet \subseteq p_1\bullet$.

Theorem 3.1 [Barkaoui,1996] *An asymmetric choice net $(\mathcal{N}, \mu_0)$ such that $\forall p \in P\ \forall t \in p\bullet$: $W(p,t) = V(p)$ for some $V : P \to \mathbb{N}$, is live if and only if for all siphons S, $\forall \mu \in \mathcal{R}(\mathcal{N}, \mu_0)\ \exists p \in S$ such that $\mu(p) \geq V(p)$.*

3.2 Conditions for Deadlock Prevention and Liveness Enforcement

Definition 3.4 *Let $\mathcal{N} = (P, T, F, W)$ be a Petri net, $\mathcal{M}$ the set of all markings of $\mathcal{N}$ and $U \subseteq \mathcal{M}$. A* **supervisory policy** *Ξ is a function $\Xi : U \to 2^T$ that maps to every marking a set of transitions that the Petri net is allowed to fire. The markings in $\mathcal{M} \setminus U$ are called* **forbidden markings**.

We denote by $\mathcal{R}(\mathcal{N}, \mu_0, \Xi)$ the set of reachable markings when $(\mathcal{N}, \mu_0)$ is supervised with Ξ. It is known that if $(\mathcal{N}, \mu_0)$ is live, then $(\mathcal{N}, \mu)$ with $\mu \geq \mu_0$ may not be live. The same is true for deadlock-freedom, as shown in figure 2. The following result shows that if liveness is enforcible at marking μ or if deadlock can be prevented at μ, then this is also true for all markings $\mu' \geq \mu$.

Proposition 3.3 *If a supervisory policy Ξ which prevents deadlock in $(\mathcal{N}, \mu_0)$ exists, then for all $\mu \geq \mu_0$ there is a supervisory policy which prevents deadlock in $(\mathcal{N}, \mu)$. The same is true for liveness enforcement.*

Proof: Let $\mu_1 \geq \mu_0$. A supervisory policy for $(\mathcal{N}, \mu_1)$ is Ξ_1 defined as follows:

$$\Xi_1(\mu + \mu_1 - \mu_0) = \begin{cases} \Xi(\mu) \cap T_f(\mu) & \text{for } \mu \in \mathcal{R}(\mathcal{N}, \mu_0) \\ \emptyset & \text{otherwise} \end{cases}$$

where $T_f(\mu)$ denotes the transitions enabled by the marking μ, apart from the supervisor. □

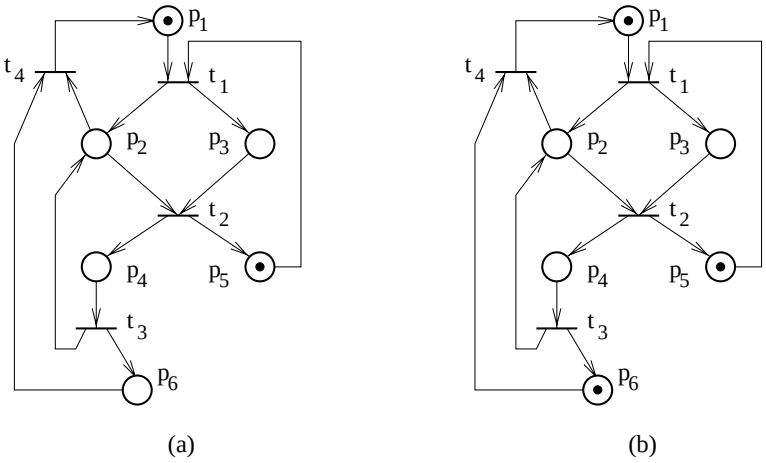


Figure 2: A Petri net that for the initial marking μ_0 shown in (a) is live, and for the initial marking $\mu \geq \mu_0$ shown in (b) is not even deadlock-free.

Definition 3.5 [Murata] *A Petri net is said to be* **(partially) repetitive** *if there is a finite marking M_0 and a firing sequence σ from M_0 such that every (some) transition occurs infinitely often in σ.*

Theorem 3.2 [Murata] *A Petri net is (partially) repetitive if and only if a vector x of positive (nonnegative) integers exists, such that $D \cdot x \geq 0$, $x \neq 0$.*

Proof: "⇒" (This part of the proof which does not appear in [Murata] is given to help the proofs of Theorem 3.3 and Corollary 3.3.) In this proof the marking is regarded as the *marking vector*. Let U be the set of transitions which appear infinitely often in an infinite firing sequence σ enabled for some finite marking M_0. We are to prove that a vector of nonnegative integers x, $x(i) \neq 0$ $\forall t_i \in U$ exists, such that $D \cdot x \geq 0$. When σ is fired, let M_0 be the initial marking, M_1 the marking reached after each transition from U fired at least once, ... M_k the marking reached after each transition from U fired at least k times.

Let V_n be a nonempty set of the form $V_n = \{y \in \mathbb{N}^n : \nexists y_i \in V_n, y \neq y_i, y \geq y_i \text{ or } y \leq y_i\}$. Next it is proved by induction that V_n is finite (i.e. it cannot have infinitely many elements). Assume that any V_{n-1} is finite. Then, let $y_{s,n} \in V_n$; $V_n \subseteq \bigcup_{k,u} C_{k,u}$, where $C_{k,u} = \{y \in \mathbb{N}^n : y(j_k) = u, y(i_k) >$

$y_{s,n}(i_k), \nexists y_i \in V_n, y \neq y_i, y \geq y_i$ or $y \leq y_i\}$, is defined for $0 \leq u < y_{s,n}(j_k)$ and $k = 1, 2 \ldots n(n-1)$ corresponds to the possiblities in which $i_k \neq j_k$, $0 \leq i_k, j_k \leq n$ can be chosen. The induction assumption implies that each $C_{k,u}$ is finite, because the component j_k of the vectors is fixed and only the remaining $n-1$ can be varied. So V_n is finite.

Let $\mathcal{M}$ be recursively constructed as follows: initially $\mathcal{M}_0 = \{M_0\}$; for all i, $\mathcal{M}_i = \mathcal{M}_{i-1} \cup \{M_i\}$ if $\nexists y \in \mathcal{M} : y \geq M_i$ or $y \leq M_i$ and else $\mathcal{M}_i = \mathcal{M}_{i-1}$. The previous paragraph showed that $\exists n_0 \in \mathbb{N}$: $\forall k > n_0$, $\mathcal{M}_k = \mathcal{M}_{n_0}$. Let $\mathcal{M} = \mathcal{M}_{n_0}$ and $\widetilde{\mathcal{M}} = \{y \in \mathbb{N}^n : \exists y_x \in \mathcal{M}, y \leq y_x\}$. Both are finite sets.

Here it is shown that $\nexists i, j$, $0 \leq i < j$, such that $M_i \leq M_j$ leads to contradiction. Assuming the contrary, $\forall k > 0$ $\exists y_x \in \mathcal{M}$ such that $M_{k+n_0} \leq y_x$ and $M_{k+n_0} \neq y_x$. If $y \in \mathbb{N}^n$, $y_x \in \mathcal{M}$ and $y_x \geq y$, then for u such that $u \not\geq y_x$ and $u \not\leq y_x$ either $y \leq u$ or both $y \not\leq u$ and $y \not\geq u$; for u such that $u \not\geq y$ and $u \not\leq y$ either $y_x \geq u$ or both $y_x \not\leq u$ and $y_x \not\geq u$. Let $\mathcal{M}^{(1)}$ be constructed in a similar way as $\mathcal{M}$, but starting from $\mathcal{M}_0^{(1)} = (\mathcal{M} \cup \{y\}) \setminus \{u \in \mathcal{M} : u \geq y\}$, where $y = M_{1+n_0}$, and using M_{n_0+i} instead of M_i for $\mathcal{M}_i^{(1)}$. For the same reason the construction ends in finitely many steps. Also, $\mathcal{M}^{(1)} \subseteq \widetilde{\mathcal{M}}$ and $\exists n_{0,1}$ such that $\forall k > 0$ $\exists y_x \in \mathcal{M}$ such that $M_{k+n_{0,1}} \leq y_x$ and $M_{k+n_{0,1}} \neq y_x$. So we can continue in the same way with $\mathcal{M}^{(2)}, \ldots \mathcal{M}^{(j)}$, also subsets of $\widetilde{\mathcal{M}}$. However these operations cannot be repeated infinitely often: $j \leq N$, where N is the cardinality of $\widetilde{\mathcal{M}}$, because $\mathcal{M}^{(j)}$ contains at least one element from $\widetilde{\mathcal{M}} \setminus \bigcup_{i=1}^{j-1} \mathcal{M}^{(i)}$. (This is so because $y \leq u$, $y \neq u$, $u \in \mathcal{M}^{(i)} \Rightarrow y \notin \mathcal{M}^{(i)}$, also $u \in \mathcal{M}^{(i)} \setminus \mathcal{M}^{(i-1)} \Rightarrow \exists v \in \mathcal{M}^{(i-1)}$: $v \geq u$, hence $\exists u \in \mathcal{M}^{(i)}$: $y \leq u$ implies $\exists v \in \mathcal{M}$: $y \leq v$.) So, $\mathcal{M}^{(j+1)}$ cannot be constructed for some j, which implies $M_{1+n_{0,j}} \not\leq u$, $\forall u \in \mathcal{M}^{(j)}$, which is contradiction.

Therefore $\exists j, k$, $j < k$, such that $M_j \leq M_k$. Let $x = q_k - q_j$. Then $M_k - M_j \geq 0 \Rightarrow D \cdot x \geq 0$, and by construction $x \geq 0$ and $x(i) > 0$ $\forall t_i \in U$.

"$\Leftarrow$" [cf. Murata] Now consider the finite firing sequence σ_1 in which we fire $x(1)$ times t_1, then $x(2)$ times t_2, and so on. Let n be the dimension of x, $X = \sum_{i=1}^{n} x(i)$ and q_i for $i \in \overline{1, X}$ the firing vectors after each transition from σ_1 is fired (note that $q_X = x$). Then the initial marking defined by $M_0(k) = \max\{0, -\min_{i \in \overline{1,X}}\{(D \cdot q_i)(k)\}\}$, $k = \overline{1, n}$, enables σ_1. Since $M_X = M_0 + D \cdot x$, and so $M_X \geq M_0$, M_X enables σ_1 too. Now it is clear that M_0 enables $\sigma = \sigma_1\sigma_1\sigma_1 \ldots$, which is an infinite sequence in which each transition t_k s.t. $x(k) \neq 0$ appears infinitely often, and so the net is (partially) repetitive. □

In general, it may not be possible to enforce liveness or to prevent deadlock in an arbitrary, given, Petri net. This may happen because the initial marking is inappropriate or because the structure of the Petri net is incompatible with the supervision purpose. The next corollary characterizes the structure of Petri nets that allow supervision for deadlock prevention and liveness enforcement, respectively. It shows that Petri nets in which liveness is enforcible are repetitive, and Petri nets in which deadlock is avoidable are partially repetitive.

Corollary 3.1 *Let $\mathcal{N} = (P, T, F, W)$ be a Petri net.*

(a) Initial markings μ_0 exist such that deadlock can be prevented in $(\mathcal{N}, \mu_0)$ if and only if $\mathcal{N}$ is partially repetitive.

(b) (cf. [Sreenivas, 1997]) *Initial markings μ_0 exist such that liveness can be enforced in $(\mathcal{N}, \mu_0)$ if and only if $\mathcal{N}$ is* repetitive.

Proof: (a) If deadlock can be avoided in $(\mathcal{N}, \mu_0)$ then μ_0 enables some infinite firing sequence σ, and by definition $\mathcal{N}$ is partially repetitive.

On the other hand, if $\mathcal{N}$ is partially repetitive, then by theorem 3.2 there is a nonnegative transition invariant x, $x \neq 0$ such that $Dx \geq 0$. Let σ_x be a firing sequence associated to a firing vector $q = x$ and let q_1 denote the firing vector after the first transition of σ_x fired, q_2 after the first two fired, and so on to $q_k = q$. If the rows of the incidence matrix D are d_1^T, d_2^T, ..., $d_{|P|}^T$, then a marking which enables σ_x is

$$\mu_0(p_i) = -\min(0, \min_{j=1\ldots k} d_i^T q_j) \quad i = 1 \ldots |P| \tag{3}$$

At least one deadlock prevention strategy exists for μ_0: to allow only the firing sequence $\sigma_x, \sigma_x, \sigma_x, \ldots$ to fire. This infinite firing sequence is enabled by μ_0 because $\mu_0 + Dx \geq \mu_0$ and μ_0 enables σ_x.

(b) The proof is similar to (a). □

Corollary 3.2 *Let $\mathcal{N} = (P, T, F, W)$ be a Petri net and D its incidence matrix. Let σ_1 and σ_2 be firing sequences and (P_1), (P_2) the two predicates below:*

(P_1) : $(\exists \sigma_1\ \exists \mu_1', \mu_1 \in \mathcal{R}(\mathcal{N}, \mu)$ *s.t.* $\mu_1[\sigma_1 > \mu_1'$ *and* $\mu_1' \geq \mu_1)$

(P_2) : $(\exists \sigma_2\ \exists \mu_2', \mu_2 \in \mathcal{R}(\mathcal{N}, \mu)$ *s.t.* $\mu_2[\sigma_2 > \mu_2',\ \mu_2' \geq \mu_2$ *and all transitions of T appear in σ_2)*

(a) Deadlock can be prevented in $(\mathcal{N}, \mu)$ if and only if (P_1) is true.

(b) Liveness can be enforced in $(\mathcal{N}, \mu)$ if and only if (P_2) is true.

(c) (i) Nonzero nonnegative integer vectors x exist such that $D \cdot x \geq 0$ and all of them have no null entries if and only if deadlock prevention enforces liveness.

(ii) Consider an arbitrary initial marking μ_0. All supervisory policies which prevent deadlock in $(\mathcal{N}, \mu_0)$ and which are more permissive than any supervisory policy which enforces liveness in $(\mathcal{N}, \mu_0)$, enforce liveness as well if and only if for all markings $\mu \in \mathcal{R}(\mathcal{N}, \mu_0)$, if (P_1) is true then (P_2) is true.

Proof: (a) If (P_1) is true, then a deadlock prevention strategy is to allow only a firing sequence that leads from μ to μ_1, and then only the infinite firing sequence $\sigma_1, \sigma_1, \sigma_1, \ldots$. Furthermore, if deadlock can be prevented, $\mathcal{N}$ is partially repetitive by Corollary 3.1(a), so $x \geq 0$ exists such that $x \neq 0$ and $Dx \geq 0$, and following the proof of Corollary 3.1(a), a marking μ can be chosen as in equation (3) for the sequence σ_x. Then (P_1) is true by taking $\mu_1 = \mu$ and $\sigma_1 = \sigma_x$.

(b) The proof is similar to (a).

(c) (i) "⇒" Let μ_0 be the initial marking and let Ξ be an arbitrary supervisory policy which prevents deadlock in $(\mathcal{N}, \mu_0)$. By part (a), (P_1) is true for all $\mu \in \mathcal{R}(\mathcal{N}, \mu_0, \Xi)$. Let x_1 be the firing vector associated to the firing sequence σ_1 from (P_1) for some marking μ that was reached. In (P_1), $\mu_1' \geq \mu_1$ implies $Dx_1 \geq 0$, so x_1 does not contain null elements. Hence σ_1 includes all transitions of the net. Because μ was arbitrary, and μ_1 reached from μ enables σ_1, this shows that for all reachable markings μ no transition is dead. So Ξ also enforces liveness.

(i) "⇐" Assume the contrary. Then there is a nonnegative integer vector x such that $Dx \geq 0$ and x has some of its elements zero. Let Ξ be a deadlock prevention policy for $(\mathcal{N}, \mu_0)$, where μ_0 is such that it enables σ_x, a transition sequence that contains $x(i)$ times each of the transitions t_i of the net. If Ξ is defined to allow only the repeated firing $\sigma_x\sigma_x\sigma_x \ldots$, then deadlock is prevented but liveness is not enforced, since σ_x does not include all transitions of the net. Contradiction.

(ii) "⇒" Assume the contrary. Then there is a supervisory policy Ξ which prevents deadlock and $\exists \mu \in \mathcal{R}(\mathcal{N}, \mu_0, \Xi)$ such that (P_1) is true and (P_2) is not. Then by part (b), $(\mathcal{N}, \mu)$ cannot be made live, so Ξ does not enforce liveness, which is a contradiction.

(ii) "⇐" Let Ξ be a supervisory policy which prevents deadlock in $(\mathcal{N}, \mu_0)$. The proof checks that for all $\mu \in \mathcal{R}(\mathcal{N}, \mu_0, \Xi)$ there is a transition sequence enabled by μ whose firing is accepted by Ξ and which includes all transitions. Let $\mu \in \mathcal{R}(\mathcal{N}, \mu_0, \Xi)$. Because deadlock is prevented, (P_2) is true since (P_1) is true. Let Ξ_L be the supervisory policy that enforces liveness in $(\mathcal{N}, \mu_0)$ by firing $\sigma\sigma'\sigma_2\sigma_2\sigma_2 \ldots$, where $\mu_0[\sigma > \mu[\sigma' > \mu_2$, and σ_2 and μ_2 are the variables from (P_2). Because Ξ is more permissive than any liveness enforcing policy, Ξ is more permissive than Ξ_L. Thus Ξ allows $\sigma'\sigma_2$ to fire from μ. Therefore all transitions appear in some firing sequence enabled by μ and allowed by Ξ. □

The important part of Corollary 3.2 is part (c), because it gives some insight about the relation between deadlock prevention and liveness enforcement. Figure 3(a) shows an example for part (c)-(i), in which all nonnegative vectors x such that $Dx \geq 0$ are a linear combination with nonnegative coefficients of $[1, 2, 1, 1]^T$ and $[2, 3, 3, 3]^T$. Figure 3(b) shows an example for part (c)-(ii) of Corollary 3.2. Indeed, all markings μ that enable any of t_1, t_2 or t_4 satisfies (P_2). Also, a marking that enables only t_3 either leads to deadlock or enables the sequence t_3, t_4 and hence satisfies (P_2). For instance, the deadlock prevention policy that repeatedly fires t_2, t_1 does not enforce liveness because it does not satisfy the requirement of Corollary 3.2(c)-(ii) to be more permissive than any liveness enforcing supervisors.

Let Ξ denote a supervisory policy. Let $\mathcal{R}(\mathcal{N}, \mu_0, \Xi)$ denote the set of reachable markings from initial marking μ_0, when $(\mathcal{N}, \mu_0)$ is supervised by Ξ.

Corollary 3.3 *Consider a Petri net $\mathcal{N} = (P, T, F, W)$ which is not repetitive. Then at least one transition exists such that for any given finite initial marking it cannot fire infinitely often. Let T_D be the set of all such transitions. There are initial markings μ_0 and a supervisory policy Ξ such that $\forall \mu \in \mathcal{R}(\mathcal{N}, \mu_0, \Xi)$, no transition in $T \setminus T_D$ is dead.*

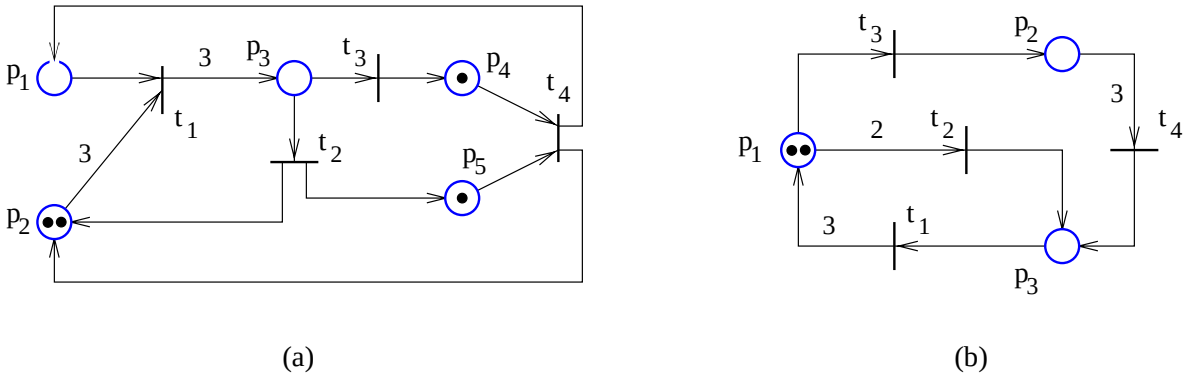


Figure 3: Examples for Corollary 3.2(c): (a) for part (i) and (b) for part (ii)

Proof: There is an integer vector $x \geq 0$ with *maximum support* such that $Dx \geq 0$, which means that for all integer vectors $w \geq 0$ such that $Dw \geq 0$, $\|w\| \subseteq \|x\|$. Indeed if $y \geq 0$, $z \geq 0$ are integer vectors and $Dy \geq 0$, $Dz \geq 0$, then $D(z+y) \geq 0$ and so $y+z \geq 0$ and $\|y\|, \|z\| \subseteq \|y+z\|$.

If $t_j \in T$ can be made live, there is a finite marking that enables an infinite firing sequence σ such that t_j appears infinitely often in σ. Therefore, using the argument from the necessity proof of Theorem 3.2 there is $y \geq 0$ such that $Dy \geq 0$ and $y(j) > 0$. Since x has maximum support, $\|y\| \subseteq \|x\|$ and so $t_j \in \|x\|$. This proves that all transitions that can be made live are in $\|x\|$. Next, the proof shows that all transitions in $\|x\|$ can be made live, which implies that T_D is nonempty and $T \setminus T_D = \|x\|$.

Let σ_x be a firing sequence associated with x, i.e. every $t_i \in T$ appears $x(i)$ times in σ_x. Then there is a marking μ_0 given by equation (3) which enables the infinite firing sequence $\sigma_x, \sigma_x, \sigma_x, \ldots$. Also, we may choose Ξ to restrict all possible firings to the former infinite firing sequence, so all transitions in $\|x\|$ can be made live. □

In Corollary 3.3, T_D is nonempty. Otherwise, since all transitions from $T \setminus T_D$ could simultaneously be made live, this would imply that $\mathcal{N}$ is repetitive, which is a contradiction. A special case is $T \setminus T_D = \emptyset$, when the Petri net is not even partially repetitive, and so deadlock can not be avoided for any finite marking.

It was already shown that only repetitive Petri nets can be made live. The corollary above shows that the set of transitions of a partially repetitive Petri net can be uniquely divided in transitions that can be made live and transitions that cannot be made live. So the liveness property of partially repetitive Petri nets is that all transitions that can be live are live.

Further on we prove an existence result for supervisors which enforce linear constraints.

Theorem 3.3 *Let $\mathcal{N}$ be a Petri net. Let Ξ be a quality like* liveness, deadlock-freedom, a.o.*, that has the property that for any marking μ_x so that Ξ can be enforced for μ_x, Ξ can be enforced for*

all markings $\mu \geq \mu_x$. If Ξ can be enforced in $\mathcal{N}$ for some markings, then $\mathcal{N}$ can be supervised by enforcing linear constraints to enforce Ξ for some markings.

Proof: The set of markings acceptable for the supervisory policy Σ enforcing Ξ is a subset of the set of markings such that Ξ holds in $\mathcal{N}$. We call μ a minimal marking accepted by Σ if there is no acceptable marking μ_i s.t. $\mu_i \leq \mu$ and $\mu_i \neq \mu$. Let $\mathcal{M}$ be the set of minimal markings accepted by Σ. We claim that $\mathcal{M}$ is finite. Assume the contrary. Let $\mu_k \in \mathcal{M}$. Then for all other markings $\mu_i \in \mathcal{M}$ there are $p_x, p_y \in P$ (P is the set of places of $\mathcal{N}$) such that $\mu_i(p_x) > \mu_k(p_x)$ and $\mu_i(p_y) < \mu_k(p_y)$. Further on, we reach a contradiction by the same reasoning as in the necessity proof of Theorem 3.2. Since $\mathcal{M}$ is finite, we may find linear constraints which enforce the condition that all reachable markings μ are in the space $\mu \geq \mu_{i_1} \vee \mu \geq \mu_{i_2} \vee \ldots \mu \geq \mu_{i_N}$, where $\mathcal{M} = \{\mu_{i_1}, \mu_{i_2}, \ldots \mu_{i_N}\}$. For example a rough solution is to use a single linear constraint given by the inequality $\mu \geq \mu_{max}$, where $\mu_{max}(p_i) = \max_{\mu_k \in \mathcal{M}} \mu_k(p_i)\ \forall p_i \in P$. □

4 The Deadlock Prevention Method

4.1 Petri Net Supervisors Based on Place Invariants

This subsection is an outline of results from [Moody, 1998] and [Yamalidou, 1996] for supervisors based on linear constraints, in the particular case of fully controllable and observable Petri nets.

The control problem considered here is to enforce a set of n_c linear constraints to prevent reaching undesired markings of a Petri net. The constraints are written in a matrix form:

$$L \cdot \mu_p \leq b \tag{4}$$

where L is an integer $n_c \times n$ matrix (n_c - the number of constraints, n - the number of places of the given Petri net), b is an integer column vector and μ_p denotes a marking vector.

Let μ_c be a vector of n_c nonnegative slack variables, defined as:

$$\mu_c = b - L \cdot \mu_p \tag{5}$$

Let μ_{c0} be the slack variables that correspond to the initial marking μ_{p0}, that is $\mu_{c0} = b - L\mu_0$. Let q be the firing vector associated with the transitions that led the Petri net from μ_{p0} to μ_p and D_p the incidence matrix, that is $\mu_p = \mu_{p0} + Dq$. So we see that $\mu_c = b - L \cdot (\mu_{p0} + D_p \cdot q)$, which also can be written as:

$$\mu_c = \mu_{c0} + (-LD_p) \cdot q \tag{6}$$

Therefore μ_c may be regarded as a marking of some additional **control places**, where the extended (supervised) Petri net has a marking vector $\mu = [\mu_p^T, \mu_c^T]^T$, and an incidence matrix $D = [D_p^T, D_c^T]^T$, and where $D_c = -LD_p$.

In the supervised net, initial markings μ_{p0} such that $L \cdot \mu_{p0} > b$ cannot be considered, since equation (5) shows that in this case μ_{c0} will not be nonnegative, and it does not make sense in

classic Petri nets to have places with negative markings. When the constraints are initially satisfied, the initial marking of the control places may be chosen according to equation (5), and therefore the constraints will remain satisfied for any reachable marking, since the D_c part of the incidence matrix prevents any firings which would attempt to make any of the variables of μ_c negative.

The way the constraints are enforced prevents only forbidden markings to be reached, so the supervisor is maximally permissive. The next theorem summarizes the construction above:

Theorem 4.1 *Let a plant Petri net with controllable and observable transitions, incidence matrix D_p and initial marking μ_{p0} be given. A set of n_c linear constraints $L\mu_p \leq b$ are to be imposed. If $b - L\mu_{p0} \geq 0$ then a Petri net controller (supervisor) with incidence matrix $D_c = -LD_p$ and initial marking $\mu_{c0} = b - L\mu_{p0}$ enforces the constraint $L\mu_p \leq b$ when included in the closed loop system $D = [D_p^T, D_c^T]^T$. Furthermore, the supervision is maximally permissive.*

Proof: See [Moody, 1998] and [Yamalidou]. □

Because $D_c = -LD_p$, every row of $[L, I]$ is a place invariant of the incidence matrix of the closed loop system, D.

4.2 Siphon Control Based on Place Invariants

Proposition 3.1 showed that in a PT-ordinary Petri net deadlock is not possible if all siphons are controlled. This suggests that all siphons should be made controlled siphons. An easy way to make a siphon controlled is to create a place invariant to control the siphon. This is done below by adding an additional place to the original Petri net. Early references of this approach for siphon control are [Barkaoui, 1995] and [Ezpeleta, 1995]. This section presents it as a special case of the supervision method based on place invariants (section 4.1). The operations described here do not depend on the fact that the structure they are applied to is a siphon, so they are described in more general terms.

Let $\mathcal{N} = (P, T, F, W)$ be a Petri net. Given a set of places S, $\sum_{p \in S} \mu(p) \geq 1$ is the desired control policy. This constraint can be enforced using the methodology of invariant based supervision of [Moody, 1998], [Yamalidou], outlined in section 4.1, which yields an additional place C, called **control place**. The place invariant created is x, such that $x(i) = 1$ for $p_i \in S$, $x(i_C) = -1$ and $x(i) = 0$ for all other indices, where i_C is the row index of C in the closed loop incidence matrix. This invariant corresponds to the equation

$$\mu(C) = \sum_{p_k \in S} \mu(p_k) - 1 \tag{7}$$

where the constant (-1) results from the initial marking of the control place. There are several particular cases:

(a) $\bullet C = \emptyset$ and $C\bullet \neq \emptyset$: no transition increases the marking of S and there are transitions which decrease the marking of S. In this case C alone makes up a minimal siphon which cannot be controlled (see also [Moody, 1998, p.87-88]).

(b) $C\bullet \subseteq \bullet S$ (in particular $C\bullet = \emptyset$): no transition can make S token free. Also, $C\bullet \subseteq \bullet S$ if and only if S is a trap. Therefore when S is also a siphon, it is (trap) controlled for all initial markings μ_0 that satisfy $\sum_{p \in S_0} \mu_0(p) \geq 1$.

(c) $\bullet C = \emptyset$ and $C\bullet = \emptyset$: the marking of S cannot vary, and so there is a place invariant x such that $x(i) = 1$ for all $p_i \in S$ and $x(i) = 0$ otherwise.

Case (a) detects transitions that cannot be made live when S is a siphon (Corollary 3.3 and Corollary 5.3). Case (b) shows the case when S does not need control. Note that the method depends only on structural properties of the Petri net. That is, it does not detect whether S does not need control for some initial markings, but it detects only the case when S does not need control for all initial markings μ_0 such that $\sum_{p \in S} \mu_0(p) \geq 1$. Therefore the method when applied to a siphon that is not a trap, but includes a trap, always produces a control place. The reason that this is correct is that there are nonzero initial markings of the siphon such that the included trap has null marking; hence the siphon is not trap controlled for such markings. Another benefit is that a control place may reveal transitions that cannot be made live (section 4.6).

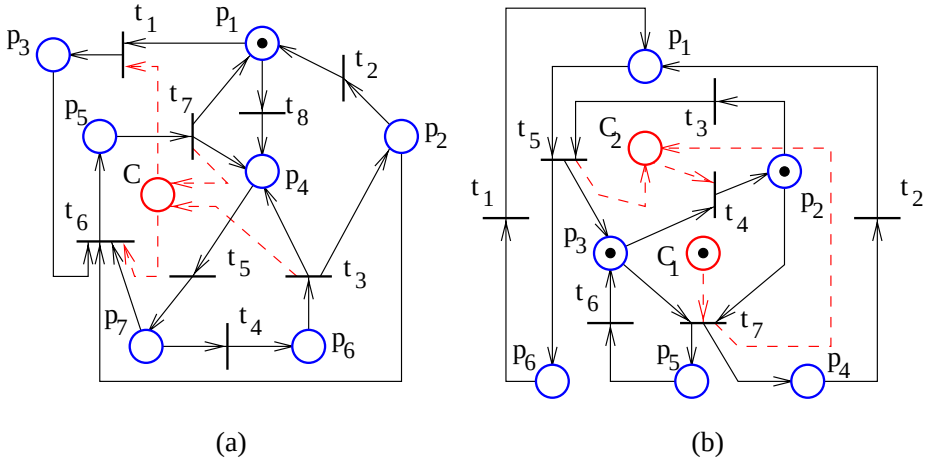


Figure 4: Siphon Control Examples. Connections of control places are dashed.

In figure 4(a) there is a single minimal siphon, $\{p_1, p_2, p_4, p_5, p_6, p_7\}$. The siphon includes a trap $\{p_4, p_5, p_6, p_7\}$, but it is not trap controlled because the marking of the trap is 0. The control place C prevents firing t_1, which would empty the siphon. In figure 4(b) the original Petri net has two minimal siphons, $\{p_3, p_2, p_5\}$ and $\{p_1, p_3, p_4, p_5, p_6\}$. Their control places are C_1 and C_2, respectively. C_1 is an example of case (a). Also, the control place C that results by controlling the minimal siphon $\{p_2, C_2\}$ satisfies $\bullet C = \emptyset$ and $C\bullet = \emptyset$.

By Theorem 4.1, the way in which the constraint $\sum_{p \in S} \mu_0(p) \geq 1$ was enforced is maximally permissive. Therefore, because the enforcement of this constraint on a siphon by definition makes the siphon controlled, there is no other more permissive way to control a siphon. This is not the only way to provide maximally permissive control of a siphon; however, any other way is equivalent. An important quality of this technique is that the closed loop remains a Petri net.

4.3 A Transformation of Petri Nets to PT-ordinary Petri Nets

Because Proposition 3.1 in section 3.1 applies to PT-ordinary Petri nets, we are interested in using a transformation to PT-ordinary Petri nets. In principle Proposition 3.2 could be used instead, but it is difficult to express its requirement in terms of linear inequalities.

We use a slightly modified form of the transformation from Lautenbach and Ridder (1996), and we call it the **PT-transformation**. Let $\mathcal{N} = (P, T, F, W)$ be a Petri net. Transitions $t_j \in T$ such that $W(p, t_j) > 1$ for some $p \in {}^\bullet t_j$ may be **split** (decomposed) in several new transitions:

> The transition t_j is **split** in $m = n(t_j)$ transitions: $t_{j,1}$, $t_{j,2}$, ... $t_{j,m}$. Also, $m - 1$ new places are added: $p_{j,1}$, $p_{j,2}$, ... $p_{j,m-1}$. The connections are as follows:
>
> (i) ${}^\bullet t_{j,1} = {}^\bullet t_j$ and $\forall p \in {}^\bullet t_{j,1}$: $W(p, t_{j,1}) = 1$
>
> (ii) $t_{j,m}{}^\bullet = t_j{}^\bullet$ and $\forall p \in t_j{}^\bullet$: $W(t_{j,m}, p) = W(t_j, p)$
>
> (iii) For $i = 2 \ldots m$, $p \in {}^\bullet t_{j,i}$ if $p = p_{j,i-1}$ or if $p \in {}^\bullet t_j$ and $j \leq W(p, t_j)$; $\forall p \in {}^\bullet t_{j,i}$: $W(p, t_{j,i}) = 1$
>
> (iv) For $i = 1 \ldots m - 1$, $p \in t_{j,i}{}^\bullet$ if $p = p_{j,i}$

The **PT-transformation** consist in splitting all transitions t such that $W(p, t) > 1$ for some $p \in {}^\bullet t$. In this way the transformed Petri net is PT-ordinary. A few properties are apparent:

$$|p_{j,i}{}^\bullet| = |{}^\bullet p_{j,i}| = 1 \qquad i = 1 \ldots m - 1 \tag{8}$$

$$|t_{j,i}{}^\bullet| = 1 \qquad i = 1 \ldots m - 1 \tag{9}$$

$$|{}^\bullet t_{j,i}| = m - i + 1 \qquad i = 1 \ldots m \tag{10}$$

A split transition t is replaced with new places and transitions and so it does not exist as an element of the set of transitions in the PT-transformed net.

Let P_T be the set of places of the transformed net. To a marking μ of the original net we associate in the transformed net a marking μ_T such that $\mu_T(p) = \mu(p)$ $\forall p \in P$ and $\mu_T(p) = 0$ $\forall p \in P_T \setminus P$.

Firing of an unsplit transition t_j in the original net corresponds to firing the same transition in the transformed net. Firing of a split transition t_j in the original net corresponds in the transformed net to firing the sequence $t_{j,1} \ldots t_{j,m}$ in which t_j was split. For similar initial markings μ and μ_T (see above) the firing sequence σ_T corresponds to a firing sequence σ, such that every split transition t_j

in σ is replaced in σ_T by its components $t_{j,1} \dots t_{j,m}$, and firing σ in $\mathcal{N}$ produces a similar marking μ' to the marking μ'_T reached by firing σ_T in the transformed net.

An example is the Petri net of figure 6, that becomes as in figure 7(a) after being PT-transformed. The transition t_2 is replaced by $t_{2,1}$ and $t_{2,2}$ and t_3 by $t_{3,1}$ and $t_{3,2}$. Firing t_2 in the original net (figure 6) corresponds to firing the sequence $t_{2,1}$, $t_{2,2}$ in the transformed net (figure 7(a)) and firing t_3 to the firing sequence $t_{3,1}$ and $t_{3,2}$.

4.4 The Idea of the Method

A deadlocked PT-ordinary Petri net has an empty siphon, by Proposition 3.1. Therefore a PT-ordinary Petri net which has all minimal siphons controlled, cannot reach deadlock.

Section 4.2 introduced a technique that controls siphons with maximal permissivity. Apparently, an approach for deadlock prevention would be to control all uncontrolled minimal siphons of a Petri net. A closer look, however, reveals that this does not guarantee that deadlock is not reached. Indeed, from Proposition 3.1 it can be deduced that a PT-ordinary Petri net with all siphons controlled is deadlock-free, but it cannot be deduced that a PT-ordinary Petri net in closed loop with a supervisor which controls its uncontrolled siphons is deadlock-free. In particular, when the maximally permissive approach presented in section 4.2 is used, one may find examples in which the control places which are added to control siphons of the Petri net create new uncontrolled siphons (for instance, such a Petri net is in figure 5(a)) and do not make the Petri net deadlock-free. The reason is not that the maximally permissive control of siphons is done through control places and not some other way, but the fact that a Petri net may have the property that a marking exists such that all siphons have a token, and all transitions enabled by it, if fired, would empty a siphon.

At this point, if the siphon control method of section 4.2 is used, a natural way to try to overcome the difficulty mentioned above is to apply the siphon control method again for the new uncontrolled siphons which were created by controlling the original uncontrolled siphons. The idea would be that eventually an improved Petri net is obtained such that all siphons are controlled. The final Petri net is the initial Petri net in closed loop with the supervisor defined by the control places that were added. The form of an iteration would be:

> Use the supervisory control method on each new minimal siphon that has at least one input transition (i.e. not a minimal siphon that is a source place). For each considered siphon S, add the control place C only if $C\bullet \not\subseteq \bullet S$.

The purpose of these iterations is to obtain a final Petri net with no uncontrolled siphons, if possible. So far the possibility that the original Petri net might have source places or that source places may appear during iterations was ignored. Each source place is by itself a minimal siphon and such a siphon cannot be effectively controlled. Indeed, the siphon control method of section 4.2 would yield another source place as a control place, and the new source place is a again an uncontrolled siphon. Corollary 5.3 shows that if such places appear, the initial net cannot be made live (is not repetitive.) Extensions to deal with the case when source places are present are discussed in section 4.6. Section 4.7 states the deadlock prevention algorithm.

Note that without splitting transitions, this iterative procedure modifies the original net only by adding new places and arcs to the transitions of the initial net. Formally, if we use $\mathcal{N}_i = (P_i, T_i, F_i, W_i)$ to denote the Petri net at the end of iteration i, for any positive integers m, n, $m < n$, $P_m \subset P_n$, $T_m = T_n$, $F_m \subset F_n$ and $W_m(x) = W_n(x)\ \forall x \in F_m$.

4.5 Implicit Inequalities

When control places are successively added by repeated application of the siphon control method of section 4.2, the set of acceptable markings of the original net is gradually restricted. In this section we show that to each (minimal) siphon corresponds a linear inequality in terms of the marking of the original net, which expresses the requirement that the siphon be controlled. For some siphons this requirement may be satisfied if it is satisfied for the initial marking, while other siphons need a control place to make sure that all reachable markings will satisfy the constraint. These constraints are important because the deadlock prevention method assumes that the initial marking satisfies them, and also restricts the set of reachable markings to the markings that satisfy them. We refer first to the case where no transition was yet split and the siphon control method was repeatedly applied for several iterations.

4.5.1 No transitions were split

Let $\mathcal{N}_i = (P_i, T_i, F_i, W_i)$ be the Petri net at the beginning of iteration i. Because no transitions were split, $P_{i+1} = P_i \cup \mathcal{C}_i$, for all i, where $\mathcal{C}_i$ is the set of control places that were added in iteration i. Any control place $C_k^{(u)}$ added at some iteration u enforces the constraint (see equation (7)) $\mu(C_k^{(u)}) = \sum_{p \in S_k^{(u)}} \mu(p) - 1$, where $S_k^{(u)}$ is the siphon controlled by $C_k^{(u)}$. Successively replacing the expressions of the markings of control places from $S_k^{(u)}$ added at previous iterations, we eventually come up with $\mu(C_k^{(u)}) = \sum_{p_i \in P_1} a_{ik}^{(u)} \cdot \mu(p_i) - c_k^{(u)}$, which expresses $\mu(C_k^{(u)})$ in terms of the marking of the places of the original Petri net $\mathcal{N}_1$, where $a_{ik}^{(u)}$, $i = 1 \ldots |P_1|$ are nonnegative integers, not all 0, and $c_k^{(u)}$ is an integer such that $c_k^{(u)} \geq u$. Since $\mu(C_k^{(u)}) \geq 0$, the inequality enforced by $C_k^{(u)}$ is:

$$\sum_{p_i \in P_1} a_{ik}^{(u)} \cdot \mu(p_i) \geq c_k^{(u)} \tag{11}$$

In the case of the siphons $S_j^{(u)}$ which do not need a control place in order for $S_j^{(u)}$ to be controlled, the only requirement is that the constraint $\sum_{p \in S_j^{(u)}} \mu(p) \geq 1$ holds true for the initial marking. In precisely the same way as before, this requirement may be written as

$$\sum_{p_i \in P_1} a_{ij}^{(u)} \cdot \mu_0(p_i) \geq c_j^{(u)} \tag{12}$$

where the coefficients have the same properties as before and μ_0 is the initial marking of the initial Petri net $\mathcal{N}_1$.

Any marking of the original net which does not satisfy one or more of the desired constraints (in the form of equations (11) and (12)) is called a **forbidden marking**.

Considering the total net, which includes all of the control places added so far, let S be a set of places (e.g. a siphon). A certain constraint, such as $\sum_{p \in S} \mu(p) \geq 1$, must be placed on S. Let Φ be the current set of forbidden markings and let Φ_S be the set of markings of the original net which do not satisfy the desired constraint. We say that S is **implicitly controlled** if $\Phi_S \setminus \Phi = \emptyset$. In other words, S is implicitly controlled when the desired constraint on S holds whenever the inequalities enforced by the previous control places hold. Therefore, if S is implicitly controlled, a control place for S is not necessary.

4.5.2 There are split transitions

Eliminating as above the markings of control places, the inequalities are in terms of markings of the places of the original net and the places that were added by splitting transitions. If P_s denotes the set of places added through transition splits, the final form of the inequalities is obtained by replacing $\mu(p) = 0$ for all $p \in P_s$. The reason in doing so is that by simulating a Petri net with a PT-transformed net, only markings μ of the latter such that $\mu(p) = 0$ for all $p \in P_s$ have a counterpart in the original Petri net (see section 4.3).

4.6 Extensions to Deal with Source Places

If the method would be based only on iterating the operations of sections 4.4 and 4.3, it would not handle efficiently Petri nets which cannot be made live for any finite marking, because in this case it is likely that source control places will appear even if the original Petri net does not have any source places. As discussed at the end of section 4.4, source places are minimal siphons which cannot efficiently be controlled by the siphon control method of section 4.2, because a control place of a source place is also a source place and hence creates a new uncontrolled minimal siphon.

It is important to consider the source places. If they are not considered for control, even if all others siphons are controlled, deadlock might still be reachable. Indeed, by Corollary 3.3 there is a nonempty set of transitions T_D with the property that any transition in T_D cannot be made live for any finite marking if the Petri net cannot be made live for any finite marking. This means that, given a finite initial marking, after a finite number of firings of transitions from T_D, all transitions of T_D are dead and the Petri net will behave as if it would be reduced by removing all transitions in T_D. Note that all transitions connected to source places appear in T_D, but not all transitions of T_D are connected to source places in the original net. Because the reduced net may have siphons which do not appear in the original Petri net, it is clear why deadlock might still be reachable if source places are ignored.

Therefore the following extensions are added to the deadlock prevention method. The method will partition the net of each iteration, which will be denoted as the **total net**, into two subnets: the **inactive subnet** and the **active subnet**. At every iteration, if one or more transitions are

detected as transitions which cannot be made live for any finite marking, they are partially moved from the *active subnet* to the *inactive subnet*. In this way any siphon structure which might appear when those transitions become dead are detected and controlled, if needed. Basically, the *inactive subnet* contains the part of the Petri net which was detected as going to deadlock for any finite marking. The *active subnet* can be made live if all transitions which cannot be made live in the *total net* have been removed from it.

At the beginning of iteration $k-1$, let $\mathcal{N}_{k,1}$, $\mathcal{N}_{k,1}^A$ and $\mathcal{N}_{k,1}^I$ denote the total Petri net and respectively its active and inactive subnets. If $\mathcal{N}_{k,1}^A$ has source places, an iterative procedure is applied to update the subnets. Iterations are necessary because removing transitions or transition arcs from the *active subnet* may produce new source places. The form of these iterations is:

1. All source places of the *active subnet* are moved to the *inactive subnet*. For every transition t in the postset of a source place, the transition t and all its arcs are moved to the *inactive subnet*. Any place p such that an arc connected to it was moved in the inactive subnet is copied there.

   The formal description of these operations is given below, where the operator $\bullet$ always is taken with respect to $\mathcal{N}_{k,i}^A$:
   $P_{k,i+1}^I = P_{k,i}^I \cup \{p \in P_{k,i}^A : \exists p' \in \bullet(\bullet p) \cup \bullet(p\bullet) \text{ s.t. } \bullet p' = \emptyset\} \cup \{p \in P_{k,i}^A : \bullet p = \emptyset\}$
   $P_{k,i+1}^A = P_{k,i}^A \setminus \{p \in P_{k,i}^A : \bullet p = \emptyset\}$
   $T_{k,i+1}^I = T_{k,i}^I \cup \{t \in T_{k,i}^A : \exists p \in \bullet t,\ \bullet p = \emptyset\}$
   $T_{k,i+1}^A = T_{k,i}^A \setminus (T_{k,i+1}^I \setminus T_{k,i}^I)$
   $F_{k,i+1}^I = F_{k,i}^I \cup \{(p,t) \in F_{k,i}^A : \exists p' \in \bullet t,\ \bullet p' = \emptyset\} \cup \{(t,p) \in F_{k,i}^A : \exists\ p' \in \bullet t,\ \bullet p' = \emptyset\}$
   $F_{k,i+1}^A = F_{k,i}^A \setminus (F_{k,i+1}^I \setminus F_{k,i}^I)$
   $W_{k,i+1}^I : F_{k,i+1}^I \to \mathbb{N}$ is defined by $W_{k,i+1}^I(x) = W_{k,i}(x)$
   $W_{k,i+1}^A : F_{k,i+1}^A \to \mathbb{N}$ is defined by $W_{k,i+1}^A(x) = W_{k,i}(x)$.
   This construction keeps the total net unchanged: $\mathcal{N}_{k,i} = \mathcal{N}_{k,i+1}$.

2. Step 1 is repeated until the *active subnet* has no source places.

The relation between the total net $\mathcal{N}_k$ and its subnets $\mathcal{N}_k^I$ and $\mathcal{N}_k^A$ is: $P_k = P_k^I \cup P_k^A$, $T_k = T_k^I \cup T_k^A$, $F_k = F_k^I \cup F_k^A$ and $W_k : F_k \to \mathbb{N}$ is given by $W_k(x) = W_k^A(x)$ for $x \in F_k^A$ and $W_k(x) = W_k^I(x)$ for $x \in F_k^I$. By construction, $F_k^I \cap F_k^A = \emptyset$, $T_k^I \cap T_k^A = \emptyset$, but P_k^I and P_k^A may not be disjoint sets.

If the total net has no source places, the active subnet is equal to the total net and the inactive subnet is empty, that is $P_1^I = \emptyset$, $T_1^I = \emptyset$, $F_1^I = \emptyset$.

4.7 The Deadlock Prevention Algorithm

Let $\mathcal{N}_0 = (P_0, T_0, F_0, W_0)$ be the initial Petri net. The first iteration begins with $\mathcal{N}_1 = (P_1, T_1, F_1, W_1)$, which is the same as $\mathcal{N}_0$ if the latter is PT-ordinary, or else it is the PT-transformed net. The methodology of section 4.6 is used to compute the initial *inactive subnet* $\mathcal{N}_1^I$ and the *active subnet*

$\mathcal{N}_1^A$. The algorithm is iterative. In every iteration inequalities of the form $\sum_{p \in S} \mu(p) \geq 1$ are enforced on sets of places S in the total net. In every iteration the *active subnet* is searched for new minimal siphons. A siphon is not considered to be *new* if it differs from an old siphon only by additional places resulted from transition split operations (Proposition 5.4). In particular, when the *active subnet* does not include a smaller siphon, the algorithm regards the whole subnet as a *minimal siphon*.

The purpose of the iterative process below is to produce two sets of linear constraints for the original net in the form $L\mu \geq b$ and $L_0\mu \geq b_0$, where L and L_0 are integer matrices with $|P_0|$ columns and b and b_0 are integer column vectors. The description of the algorithm iteration is given below:

The current iteration (let it be number k) modifies the nets resulted in the previous iteration: $\mathcal{N}_k^I$, $\mathcal{N}_k^A$ and $\mathcal{N}_k$.

1. If no new minimal siphon of the *active subnet* is found in the *total net*, the algorithm terminates. (In the first iteration every siphon is considered to be new.) Otherwise it continues with the next step.

2. For every new minimal siphon S of the *active subnet* the supervisory control method of section 4.2 is used to add the invariant needed to enforce $\sum_{p \in S} \mu(p) \geq 1$ in the *total net*. Let C be the control place which would result and $l\mu \geq c$ the inequality in terms of the marking of the original Petri net which is associated to the requirement that $\sum_{p \in S} \mu(p) \geq 1$ (see section 4.5). There are two cases:

   (a) the methodology of section 4.2 would yield $C\bullet \subseteq \bullet S$. In this case S does not need supervision and C is not added to the *total net*. The linear constraint (l, c) is included in (L_0, b_0).

   (b) the methodology of section 4.2 would yield $C\bullet \not\subseteq \bullet S$. In this case the place C is added to the *total net* according to the method of section 4.2. The linear constraint (l, c) is included in (L, b). The two subnets are updated as follows:

   (i) The *active subnet:* C is added to the set of places. All transition arcs of the form (C, t) and (t, C) such that $t \in T_k^A$ are copied in the set of transition arcs.

   (ii) The *inactive subnet:* the arcs of the *total subnet* which were not copied in the *active subnet* are copied in the set of transition arcs and C in the set of places.

3. The two subnets are updated as shown in section 4.6, because source places may have appeared in the *active subnet* in the previous step.

4. If the *active subnet* is no longer PT-ordinary, the transitions of the *active subnet* which do not comply with this requirement are *split* (section 4.3). The final nets of iteration k are denoted by $\mathcal{N}_{k+1}^I$, $\mathcal{N}_{k+1}^A$ and $\mathcal{N}_{k+1}$. The algorithm proceeds with the next iteration.

After the iterative process above terminates, a method that removes redundant constraints may be used to simplify (L, b). The inequalities given by (L, b) (in terms of markings of the original net $\mathcal{N}_0$) are enforced on $\mathcal{N}_0$ with the invariant based methodology of section 4.1. For all initial markings μ_0, such that $L\mu_0 \geq b$ and $L_0\mu_0 \geq b_0$, deadlock prevention in the closed loop Petri net is guaranteed for the condition of Theorem 5.2(d). The difference between the constraints (L, b) and (L_0, b_0) is that (L, b) need to be enforced by supervision, while (L_0, b_0) need not. (L_0, b_0) are guaranteed by the structure of the original Petri net in closed loop with the supervisor enforcing (L, b) for all initial markings μ_0 of the original Petri net that satisfy $L_0\mu_0 \geq b_0$ in addition to $L\mu \geq b$. The algorithm is allowed to start with initial constraints of the type (L_0, b_0), to which it may add other constraints, as necessary. However, without reducing the generality (see section 5.3.1), no initial constraints of the form (L, b) are allowed.

4.8 A Special Case: Repetitive Petri Nets

Repetitive Petri nets have the property that finite markings exist such that liveness is enforcible via supervision. So they do not have source places and source places cannot appear in the iterations of the deadlock prevention algorithm (Corollary 5.3). Because of this there is no need to partition the Petri net into the two subnets, because the *active subnet* always is equal to the *total net* and the *inactive subnet* always is empty. In this case the description of the algorithm is simplified by removing parts (i) and (ii) from step 2(b) and step 3.

The deadlock prevention algorithm simplified for repetitive Petri nets is similar to the algorithm from [Lautenbach, 1996]. A difference is that in [Lautenbach, 1996] the supervised Petri net is built by collapsing transitions produced through splitting. This operation is equivalent in our method to enforcing the set of linear inequalities. The approach we use better suits our purpose not to assume that the initial marking is known. In our approach, the requirements on the initial marking appear clearly stated: $L\mu_0 \geq b$ and $L_0\mu_0 \geq b_0$. Another difference is that the transformation to PT-ordinary Petri nets is used in our approach in a slightly simplified form, which reduces the number of siphons.

4.9 Illustrative by Examples

Example 4.1 We consider the Petri net of figure 5(a), which is repetitive. Indeed, the marking reached by firing the sequence $t_2, t_1, t_4, t_5, t_4, t_2, t_3, t_4$ is equal to the initial marking.

The original net has two minimal siphons: $\{p_1, p_2\}$ and $\{p_3, p_4\}$. The method of section 4.2 yields two control places for each of the two siphons, C_1, C_2 respectively, and $C_1\bullet \neq \emptyset$, $C_2\bullet \neq \emptyset$. The net after the first iteration is shown in figure 5(b). If μ_0 is the initial marking and if the initial markings chosen for C_1 and C_2 are $\mu_0(C_1) = \mu_0(P_1) + \mu_0(P_2) - 1$, and $\mu_0(C_2) = \mu_0(P_3) + \mu_0(P_4) - 1$, then for any reachable marking μ:

$$\mu(C_1) = \mu(P_1) + \mu(P_2) - 1 \quad (13)$$

$$\mu(C_2) = \mu(P_3) + \mu(P_4) - 1 \quad (14)$$

As shown in section 4.2, the previous two equations correspond to the enforcement of the constraints $\mu(P_1) + \mu(P_2) \geq 1$ and $\mu(P_3) + \mu(P_4) \geq 1$. The matrices L and b at the end of the iteration below reflect equations (13) and (14).

$$L = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

At the second iteration the only new minimal siphon is $\{C_1, C_2\}$. Using the equations (13),(14), the constraint $\mu(C_1) + \mu(C_2) \geq 1$ is equivalent to $\mu(P_1) + \mu(P_2) + \mu(P_3) + \mu(P_4) \geq 3$. The siphon $\{C_1, C_2\}$ is not implicitly controlled by C_1 and C_2, since $\mu(P_1) + \mu(P_2) \geq 1$ and $\mu(P_3) + \mu(P_4) \geq 1$ do not imply $\mu(P_1)+\mu(P_2)+\mu(P_3)+\mu(P_4) \geq 3$. Using again the method of section 4.2 for $\{C_1, C_2\}$, we get a new control place C_3 with initial marking $\mu_0(C_3) = \mu_0(C_1) + \mu_0(C_2) - 1$, which implies that for any reachable marking μ:

$$\mu(C_3) = \mu(P_1) + \mu(P_2) + \mu(P_3) + \mu(P_4) - 3 \tag{15}$$

The resulting net (figure 5(c)) has no new minimal siphons, therefore the algorithm terminates. The matrices L and b after the second iteration are:

$$L = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad b = \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix}$$

Because L and b cannot be simplified, the supervised net for deadlock prevention is the same as net as that of figure 5(c). By Theorem 5.2, the supervised Petri net is deadlock-free for all initial markings μ_0, such that $L\mu_0 \geq b$. In this example the matrices L_0 and b_0 are empty. □

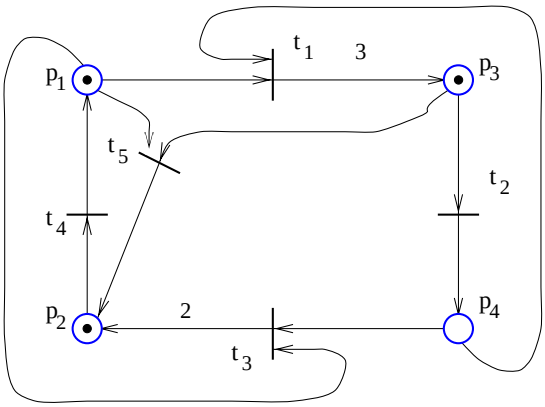
Example 4.2 Consider the Petri net of figure 6, which is not PT-ordinary. Three transitions cannot be made live, for any finite marking: t_1, t_2, t_3. At the beginning, because no source places are present, the active subnet is equal to the PT-transformed net (figure 7(a)), while the inactive subnet is empty.

The first iteration begins with the PT-transformed net. There is a single minimal siphon, $\{p_1, p_2, p_3\}$. A control place C_1 is added to the total net (figure 7(e)). Because C_1 is a source place, at the end of the iteration there is a nonempty inactive subnet. The active and inactive subnets are shown in figure 7(b) and (c). The inequality associated with C_1 is $\mu(p_1) + \mu(p_2) + \mu(p_3) \geq 1$, so at the end of this iteration $L = [1, 1, 1]$ and $b = 1$.

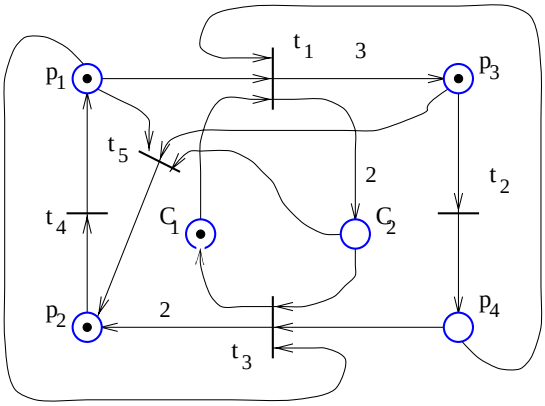
In the second iteration the active subnet has a single siphon, $\{p_1, p_2\}$. The siphon is uncontrolled, since $\mu(p_1) + \mu(p_2) \geq 1$ is not implied by $\mu(p_1) + \mu(p_2) + \mu(p_3) \geq 1$. The control place C_2 which is added is also a source place. At the end of the iteration, we have the same active subnet (figure 7(a)) and a different inactive subnet (figure 7(d)). Then the algorithm terminates, since there is no new minimal siphon in the active subnet. The matrices L and b after iteration two are:

$$L = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \qquad b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

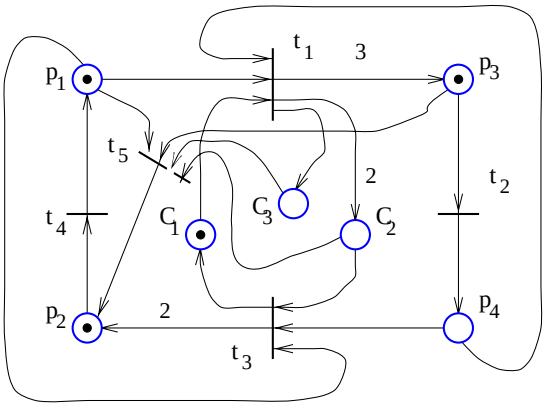
which can be simplified to $L = [1, 1, 0]$ and $b = 1$. The supervised net is shown in figure 8. By Theorem 5.2 it is deadlock-free for all initial markings μ_0 such that $L\mu_0 \geq b$. □



(a)



(b)



(c)

Figure 5: Example 4.1: (a) the original net, (b) after one iteration, (c) the final net. C_1 is a control place for the siphon $\{p_1, p_2\}$, C_2 for $\{p_3, p_4\}$ and C_3 for $\{C_1, C_2\}$.

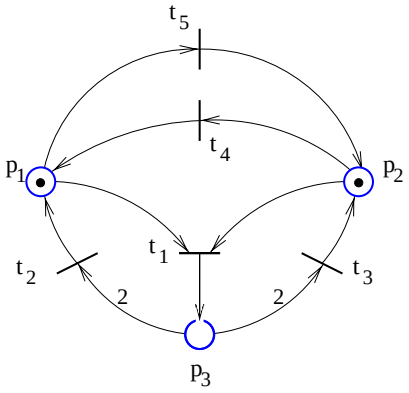


Figure 6: The initial Petri net of Example 4.2

5 Properties

5.1 Basic Properties of the Method

5.1.1 Introduction and Notations

In the deadlock prevention algorithm, we start with a Petri net $\mathcal{N}_0 = (P_0, T_0, F_0, W_0)$ that may not be PT-ordinary. New Petri nets $\mathcal{N}_i = (P_i, T_i, F_i, W_i)$, $i \geq 1$, are derived in the iterative process. The only operations of an iteration that modify the structure of the total net are the addition of a new control place (section 4.2) and transition split (section 4.3).

Adding control places does not modify the set of transitions. The set of places is increased by the set of new control places, and the set of transition arcs by the new arcs which connect the control places to already existing transitions. The old arcs have unmodified weights; new arcs connecting the new control places may have weights greater than one. If a weight of an arc entering a transition is greater than one, the Petri net is not PT-ordinary and transitions not conforming to the requirement may be split.

When a transition is split, it is replaced by a string of places and transitions. The transition that was split does not appear in the modified Petri net, and firing the old split transition is now equivalent to firing the sequence of transitions that replaced it. Let T_R be the set of transitions of the modified net that appeared by splitting and T_S the set of transitions that were split. Also, let P_R be the set of places generated by transition split. Then, $P_i = P_0 \cup P_R \cup \mathcal{C}$ and $T_i = (T_0 \setminus T_S) \cup T_R$, where $\mathcal{C}$ is the set of control places that were added.

If $t \in T_i$ and in iteration i it is split, and so $t \notin T_{i+1}$, then $\sigma_{i,i+1}(t)$ will denote the replacing sequence of transitions. If t is not split, then $\sigma_{i,i+1}(t) = t$. If $x = t_1 \dots t_k$ is a sequence of transitions,

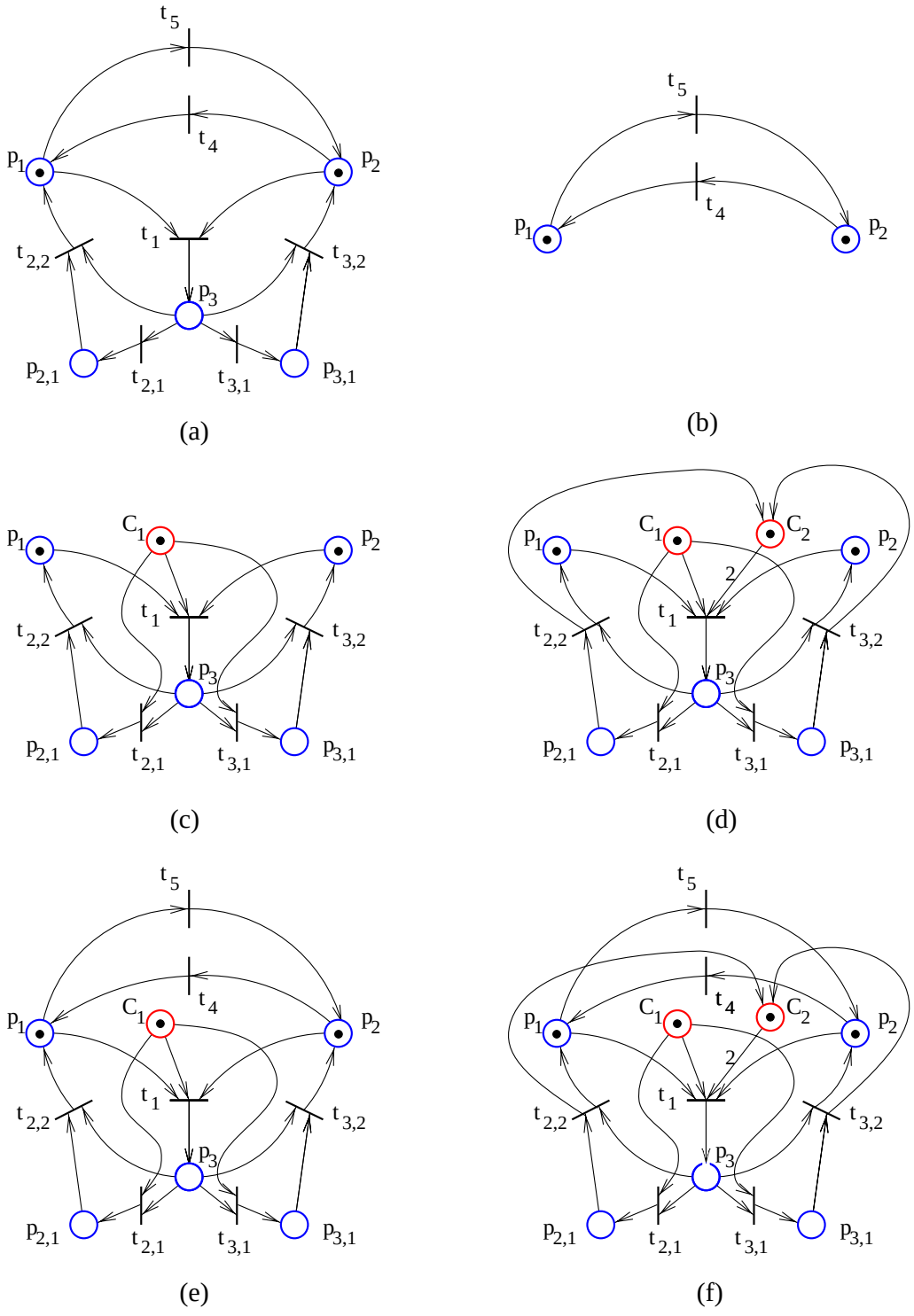


Figure 7: Example 4.2: (a) the active subnet at the beginning of iteration one; (b) the active subnet after the first iteration, which remains the same after the second iteration; (c), (d) the inactive subnet after the first iteration and the second iteration, respectively; (e) the total net after the first iteration; (f) the total net after the second iteration.

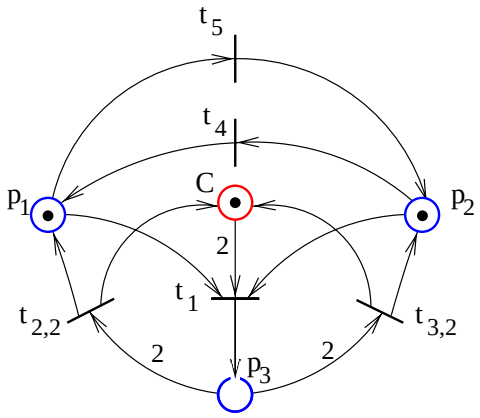


Figure 8: The Petri net of Example 4.2, supervised for deadlock-freedom

$\sigma_{i,i+1}(x)$ is the sequence that enumerates the sequences of $\sigma_{i,i+1}(t_1) \ldots \sigma_{i,i+1}(t_k)$. So:

$$\sigma_{i,i+1}(x) = \begin{cases} x & \text{if } x \in T_{i+1} \cap T_i \\ \text{replacement sequence} & \text{if } x \in T_i \setminus T_{i+1} \\ \sigma_{i,i+1}(t_1) \ldots \sigma_{i,i+1}(t_k) & \text{if } x = \{t_j\}_{j=1\ldots k},\ t_j \in T_i \text{ for } j = 1 \ldots k \end{cases}$$

where $T_i \setminus T_{i+1}$ corresponds to the set of transitions which were split, and $T_{i+1} \cap T_i$ to the set of transitions which were not split in iteration i.

Another notation is $\sigma_{k,i}(t)$, which accounts for all transitions split in iterations $1, 2 \ldots i-1$. If t was split in iteration $k \leq j < i$, then $\sigma_i(t)$ is the replacement of t in $\mathcal{N}_i$, and not in $\mathcal{N}_{j+1}$, which is $\sigma_{j+1}(t)$. The difference is that some transitions which appeared in $\sigma_{j+1}(t)$ may have been split from iteration $j+1$ to $i-1$, and thus $\sigma_i(t)$ uses their replacement. This can be written as follows

$$\sigma_{k,i}(x) = \sigma_{i-1,i}(\sigma_{i-2,i-1}(\ldots \sigma_{k,k+1}(x)\ldots))$$

where x is a transition or a sequence in T_k. In particular, $\sigma_{0,i}$ considers all transitions which were split beginning with the original Petri net $\mathcal{N}_0$.

The notations of Petri nets which are used are: $\mathcal{N}_0 = (P_0, T_0, F_0, W_0)$ – the initial Petri net, $\mathcal{N}_1 = (P_1, T_1, F_1, W_1)$ – $\mathcal{N}_0$ PT-transformed, $\mathcal{N}_i = (P_i, T_i, F_i, W_i)$ – the Petri net produced by iteration $i-1$ for $i \geq 2$, $\mathcal{N}_i^I = (P_i^I, T_i^I, F_i^I, W_i^I)$ – the inactive subnet of $\mathcal{N}_i$ and $\mathcal{N}_i^A = (P_i^A, T_i^A, F_i^A, W_i^A)$ – the active subnet of $\mathcal{N}_i$.

5.1.2 Properties

Proposition 5.1 *Let* $\mathcal{N}_k^I$, $\mathcal{N}_k^A$ *and* $\mathcal{N}_k$ *be the* inactive subnet, the active subnet *and the* total subnet *after iteration number* $k-1$.

(a) $P_k \subseteq P_{k+1}$ *for all* $k \geq 0$.

(b) Any $p \in P_k^I \setminus P_k^A$ *has in* $\mathcal{N}_k$ *the property that* $\bullet p \subseteq T_k^I$.

(c) Consider the step 2 of an iteration and let C *be a control place added to the* total net *with regard to a minimal siphon* S *of the* active subnet, $\mathcal{N}_k^A$. *Then* S *is controlled by* C *in the* active subnet *(considered as an independent net.)*

Proof: (a) By construction, control places are added to the total net and new places may be created by transition split. In this way $P_{k+1} = P_k \cup \mathcal{C}_k \cup P_{S,k}$, where $\mathcal{C}_k$ is the set of control places added in iteration k and $P_{S,k}$ is the set of places resulted from transition split in iteration k.

(b) Immediate consequence of the construction from section 4.6.

(c) If $\bullet C$ and $C\bullet$ are considered in the *total net*, the arcs of the form (C,t) and (t,C) of the *total net* which would appear in the *active subnet* are limited to (C,t) with $t \in C\bullet \cap T_k^A$ and (t,C) with $t \in \bullet C \cap (T_k^A \setminus T_k^I)$ ($T_k^A \cap T_k^I$ can only be sink transitions in the *active subnet*). Since these arcs (C,t) and (t,C) are also obtained by applying the methodology of section 4.2 directly in $\mathcal{N}_k^A$, it follows that C is a control place for S in the *active subnet*. □

In the previous proposition note that part (b) considers the *active subnet* before the update that the algorithm makes at step 3; C may remain or not in the *active subnet* after step 3, depending on whether or not C is not a source place in the *active subnet*.

Several properties also related to transition splitting are given in the next two propositions.

Proposition 5.2 *Let* $t_x \in T_0$ *and* C *be a control place added before some iteration* $m > 1$. *Assume that in the iteration number* $m-1$ t_x *is split,* $\sigma_{0,m}(t_x) = t_{x,1}, t_{x,2}, \ldots t_{x,k}$ *and the places in the replacing sequence are* $p_{x,1}, p_{x,2} \ldots p_{x,k-1}$.

(a) If $C \in \bullet t_{x,i}$ *then* $C \in \bullet t_{x,1}$.

(b) If $C \in t_{x,i}\bullet$ *then* $i = k$.

(c) $\bullet t_{x,i} \setminus \{p_{x,i-1}\} \subseteq \bullet t_{x,1}$ *for all* $1 < i \leq k$.

Proof: The proof is by induction. Assuming the properties (a) and (b) to be true for all the control places added so far (let their set be $\mathcal{C}$), let C be a control place that is to be added now with regard to a minimal siphon S of the active subnet $\mathcal{N}^A$. Note that the induction assumption implies the property (c) to be currently true: $(\bullet t_{x,i} \setminus \{p_{x,i-1}\}) \cap \mathcal{C} \subseteq \bullet t_{x,1} \cap \mathcal{C}$, and since $(\bullet t_{x,i} \setminus \{p_{x,i-1}\}) \cap P_0 \subseteq \bullet t_{x,1} \cap P_0$ and $(\bullet t_{x,i} \setminus \{p_{x,i-1}\}) \cap P_S = \emptyset$, where P_S is the set of places resulted through transition split, the property (c) is currently true: $\bullet t_{x,i} \setminus \{p_{x,i-1}\} \subseteq \bullet t_{x,1}$ $\forall 1 < i \leq k$. If $p_{x,i} \in S$, since S is minimal, $p_{x,i} \bullet \bullet \cap S \neq \emptyset$. If $i < k-1$, $p_{x,i} \bullet \bullet = p_{x,i+1}$ (see section 4.3 and note that successive transition split does not affect this property.) This implies that if $p_{x,i} \in S$ then $p_{x,j} \in S$ $\forall j = i \ldots k-1$. If C is to control $t_{x,i}$ (i.e. an arc $(C, t_{x,i})$ is to be added) then $\bullet t_{x,i} \cap S \neq \emptyset$

and firing $t_{x,i}$ reduces the number of tokens of S. If $p_{x,i-1} \notin S$ then $p_{x,1} \notin S$ and by property (c) firing $t_{x,1}$ reduces the marking of S, so C will be in $\bullet t_{x,1}$. If $p_{x,i-1} \in S$ then $p_{x,i} \in S$ and so $(\bullet t_{x,i} \setminus \{p_{x,i-1}\}) \cap S \neq \emptyset$. Further on, using relation (c), $t_{x,1}$ reduces the marking of S if $p_{x,1} \notin S$, which is true, because $p_{x,1} \in S$ would imply that S is not minimal ($S \setminus \{p_{x,1} \dots p_{x,i-1}\}$ would be a siphon.) This shows that relation (a) holds true after C is added.

If $t_{x,j}$, for $j < k$, would increase the number of tokens of S, then $p_{x,j} \in S$, because $p_{x,j} = t_{x,j}\bullet$. Then $t_{x,j} \in T^A$, because otherwise $p_{x,j}$ would be a source place in the active subnet, which is not possible. Because S is a siphon in $\mathcal{N}^A$, $\bullet t_{x,j} \cap S \neq \emptyset$, so $t_{x,j}$ cannot increase the marking of S. Therefore property (b) holds true for C. □

Proposition 5.3 *For every iteration index i:*

(a) If $P_i^A \cap P_0 = \emptyset$ then $\mathcal{N}_i^A$ is empty.

(b) Let $t \in T_0$. If $t_x \in \sigma_{0,i}(t)$ and $t_x \in T_i^A$ then every transition of $\sigma_{0,i}(t)$ is in T_i^A.

(c) Let $\mathcal{C}$ be the set of control places of $\mathcal{N}_i$, that is all the control places which were added in iterations $1, 2, \dots i-1$. There is no siphon S of the total net or of the active subnet such that $S \subseteq P_i \setminus (P_0 \cup \mathcal{C})$.

Proof: (a) $P_i^A \cap P_0 = \emptyset \Rightarrow T_0 \cap T_i^A = \emptyset$. Also, for any original transition $t_0 \in T_0$ subsequently split in $\sigma_{0,i}(t_0) = t_{0,1}t_{0,2} \dots t_{0,k}$, $t_{0,1} \notin T_i^A$. This implies that the first place $p_{0,1}$ from the split replacement is a source place, so $p_{0,1} \notin P_i^A$, which in turn implies $t_{0,2} \notin T_i^A$. Iterating in this way, no transition of $\sigma_{0,i}(t_0)$ is in T_i^A. Then $T_i^A = \emptyset \Rightarrow P_i^A = \emptyset$ and $F_i^A = \emptyset$ (section 4.6.)

(b) Assume that $\exists t_y \in \sigma_{0,i}(t)$ and $t_y \notin T_i^A$. Then $\bullet t_y \cap (P_i^I \setminus P_i^A) \neq \emptyset$ (section 4.6). By Proposition 5.2(c) $\bullet t_1 \cap (P_i^I \setminus P_i^A) \neq \emptyset$, so $t_1 \notin T_i^A$, where t_1 is the first transition of $\sigma_{0,i}(t)$. However, as in the proof of (a), this implies that all transitions of $\sigma_{0,i}(t)$ are not in T_i^A, contradicting $t_x \in T_i^A$.

(c) Let P_S be the set of places resulted from transition split: $P_S = P_i \setminus (P_0 \cup \mathcal{C})$. The proof is a direct consequence of the splitting method (section 4.3). Thus, $p \in P_S$ cannot be a source place in the total net, while the active subnet cannot anyway have source places. Further on, if P_{Sx} is the set of places from the replacement of $t_x \in T_0$ in $\mathcal{N}_i$, there are no cyclic structures only made up of places in P_{Sx}. Also, because $(\bullet\bullet P_{Sx} \setminus P_{Sx}) \cap P_S = \emptyset$ and $(P_{Sx}\bullet\bullet \setminus P_{Sx}) \cap P_S = \emptyset$ there is no cyclic structure only made up of places in P_{Sx} and other places from P_S. The same justification also applies to the active subnet, in which a replacement sequence is either completely included or not present (part (b).) □

It is interesting to find out what a siphon controlled with a control place becomes when one or more of its transitions are split. A transition t may be split after a control place C is added in the preset of t and $W(C,t) > 1$. The following proposition considers some of the effects of splitting transitions.

Proposition 5.4 *Let S be a minimal siphon in a PT-ordinary net. Assume that after adding some control places the net is no longer PT-ordinary. If a transition t is split, let P_s be the set of places generated through this split.*

(a) If before the split $t \in {}^\bullet S$, then S is no longer a siphon and $S' = S \cup P_s$ is a minimal siphon.

(b) If $t \notin {}^\bullet S$, S is still a minimal siphon.

Proof: (a) Consider that t is split in $t_{1,1}, t_{1,2} \dots t_{1,k}$ and that the new places which result are $p_{1,1}, p_{1,2} \dots p_{1,k-1}$ (the same notations as in section 4.3.) Initially $t \in {}^\bullet S \Rightarrow t \in S^\bullet$, but now $t_{1,k} \in {}^\bullet S$ and $t_{1,1} \in S^\bullet$. Since S was a minimal siphon in a PT-ordinary net, $t_{1,i} \notin S^\bullet \; \forall i > 1$ (t was split because one or more control places C were connected to t such that $W(C,t) > 1$). It follows that S is not a siphon ($t_{1,k} \notin S^\bullet$) and S' is a minimal siphon.

(b) This is obvious, since splitting t does not modify ${}^\bullet S$, but only $S^\bullet \setminus {}^\bullet S$ at most, when $t \in S^\bullet$. □

Proposition 5.5 *Let S be a siphon of $\mathcal{N}_i^A$ controlled in step 2 of iteration i with the control place C. Let P_R be the set of places resulted through transition split in iterations i to j and μ_0 be a marking of $\mathcal{N}_j$ such that $\mu_0(p) = 0 \; \forall p \in P_R$ and $\mu_0(C) = \sum_{p_i \in S} \mu_0(p_i) - 1$. For all markings μ reachable from μ_0 and such that $\mu(p) = 0 \; \forall p \in P_R$, $\mu(C) = \sum_{p_i \in S} \mu(p_i) - 1$ is satisfied.*

Proof: This is a direct consequence of the fact that C initially enforced $\sum_{p \in S} \mu(p) \geq 1$ on S and that firing an entire split replacement sequence modifies the marking of the original places in the same way as firing the transition which was split (see section 4.3.) □

Proposition 5.6 *Let $S \subseteq P_i$, $i \geq 1$. New control places are added in iteration i, and so the net resulting after the step 2 of iteration i may not be PT-ordinary. Assume that the marking constraint $\sum_{p \in S} \mu(p) \geq 1$ is currently enforced using the control place C, added in this iteration or on a previous iteration. Consider that a transition $t \in {}^\bullet S \cup S^\bullet$ is split, P_s is the set of places generated through the split, and $S' = S \cup P_s$.*

(a) If initially the arc (C,t) does not exist or if it does, $W(C,t) = 1$, then after t is split

(i) C insures in S' that $\sum_{p \in S'} \mu(p) \geq 1$ if originally $t \in {}^\bullet S$.

(ii) C insures in S that $\sum_{p \in S} \mu(p) \geq 1$ if originally $t \notin {}^\bullet S$.

(b) If initially $W(C,t) > 1$, then after t is split

(i) C insures in S' that $\sum_{p \in S'} \mu(p) \geq 1$ if originally $t \in {}^\bullet S$.

(ii) C does not insure in S that $\sum_{p\in S}\mu(p)\geq 1$ if originally $t\in S\bullet\setminus\bullet S$.

Proof: Consider that t is split in $t_{1,1},t_{1,2}\dots t_{1,k}$ and that the new places which result are $p_{1,1},p_{1,2}\dots p_{1,k-1}$ (the same notations as in section 4.3.)

Splitting t does not affect C controlling the other transitions that by firing could empty S. Consider the case (i), $t\in\bullet S$. If $t_{1,1}\notin S\bullet$, then firing any of $t_{1,j}$ does not reduce the marking of S, so $\sum_{p\in S}\mu(p)\geq 1$ remains always true. If $t_{1,1}\in S\bullet$ then by firing $t_{1,1}$ S looses one or more tokens, but $p_{1,1}$ gets one, so S' does not become empty. Also, by firing any of $t_{1,j}$, $j<k$, always is a place of P_s with a token, so S' is not empty. By firing $t_{1,k}$, because $t_{1,k}\in\bullet S$, S gets one or more tokens. So, in all cases S' cannot become empty.

Consider the case (ii), $t\notin\bullet S$, but $t\in S\bullet$. Then an arc (C,t) should exist before the split, since t takes tokens from S without returning any back. If $W(C,t)=1$, S looses one token by firing $t_{1,1}$ and no tokens by firing any other $t_{1,j}$. Because $t_{1,1}$ is enabled when C has at least one token, and so S at least two, S is not emptied by $t_{1,1}$ (or any of $t_{1,j}$.) If $W(C,t)>1$, S looses $W(C,t)$ tokens by firing $t_{1,1}$ and no tokens by firing any other $t_{1,j}$. However C looses only one token by firing $t_{1,1}$, and $W(C,t)$ tokens in total after firing all $t_{1,i}$. (After t is split, no place p exists such that $W(p,t_{1,j})>1$, for any j.) So if S has $W(C,t)$ tokens (and so C has $W(C,t)-1>0$ tokens) such that $t_{1,1}$ is enabled, by firing $t_{1,1}$, S is emptied. □

Proposition 5.7 *Let $S\subseteq P_i^A$ such that $\sum_{p\in S}\mu(p)\geq 1$ is insured for all markings reachable from a set of markings $\mathcal{M}$ of $\mathcal{N}_i^A$. Let S' be a minimal siphon of $\mathcal{N}_k^A$, $k>i$, such that $S\subseteq S'$ and $S'\subseteq S\cup P_R$, where P_R is the set of places resulted by transition split in iterations i through $k-1$. Then S' is a controlled siphon of $\mathcal{N}_k^A$, that is $\sum_{p\in S'}\mu(p)\geq 1$ for all markings μ reachable from markings μ_0 such that $\mu_0(p)=0\ \forall p\in P_R$, $\mu_0(p)=\mu_i(p)\ \forall p\in P_k^A\cap P_i^A$ and $\mu_i\in\mathcal{M}$.*

Proof: Transitions $t\in T_i$ are considered in what follows only if $\sigma_{i,k}(t)$ is contained in T_k^A. (In view of Proposition 5.3(b), a replacement sequence is either completely included in the active subnet or does not appear at all there.) Let $\sigma_{i,k}(t)=t_{1,1}t_{1,2}\dots t_{1,u}$ and $p_{1,1},p_{1,2}\dots p_{1,u-1}$ be the places resulted from the split and $Z=P_k^A\cap P_i^A$. For all $t\in T_i$ such that $\sigma_{i,k}(t)$ is in T_k^A, $\bullet t\subseteq Z$ in $\mathcal{N}_i$. Indeed, if $\bullet t\cap(P_i\setminus Z)\neq\emptyset$ then $\sigma_{i,k}(t)$ is in T_k^I, and so not in T_k^A, since $P_i\setminus Z$ are places of the inactive subnet $\mathcal{N}_k^I$.

For markings μ' reachable from $\mu_0'\in\mathcal{M}$ consider the markings μ of $\mathcal{N}_k^A$ such that $\mu(p)=0$ $\forall p\in P_k^A\cap P_R$ and $\mu(p)=\mu'(p)\ \forall p\in Z$. As a split transition property, firing t in $\mathcal{N}_i^A$ and $\sigma_{i,k}(t)$ in $\mathcal{N}_k^A$ produces the same marking change for the places of Z in $\mathcal{N}_i^A$ and $\mathcal{N}_k^A$ and if μ enables $\sigma_{i,k}(t)$ then the similar marking μ' of $\mathcal{N}_i^A$ enables t. Let t be a transition such that $\sigma_{i,k}(t)$ is contained in T_k^A:

(a) If $t\in S\bullet\setminus\bullet S$ in $\mathcal{N}_i$ then $t_{1,1}\in S'\bullet$ and $t_{1,j}\notin S'\bullet$ for all other transitions of $\sigma_{i,k}(t)$, since S' is minimal. (Otherwise all $t_{1,j}\in\bullet S'$, the last transition of $\sigma_{i,k}(t)$ satisfies in addition $t_{1,u}\in\bullet(S'\setminus P_R)$,

and so $t \in \bullet S$ in $\mathcal{N}_i$ is inferred, which is a contradiction.) Because $t_{1,1}$ cannot be enabled unless t is enabled and firing t in $\mathcal{N}_i^A$ and $\sigma_{i,k}(t)$ in $\mathcal{N}_k^A$ produces the same marking change for the places of Z in $\mathcal{N}_i^A$ and $\mathcal{N}_k^A$, $\sigma_{i,k}(t)$ cannot empty S': there is at least one token left in $S' \cap Z = S \cap Z$.

(b) If $t \in S\bullet \cap \bullet S$ in $\mathcal{N}_i^A$ then $\sigma_{i,k}(t)$ is contained in $\bullet S'$ (for all $p_{1,j}$, $p_{1,j} \in S'$). So each time one of $t_{1,j}$ fires, there is at least one token left in S'.

The case $t \in \bullet S \setminus S\bullet$ in $\mathcal{N}_i^A$ was not considered, since considering $\sigma_{i,k}(t)$ to be contained in T_k^A contradicts that S' is a siphon, because $t_{1,u} \in \bullet S'$, $t_{1,j} \notin S\bullet$ for all $1 \leq j \leq u$, $\bullet t_{1,1} \notin P_R$ and $\bullet t_{1,j} \setminus \{p_{1,j-1}\} \notin P_R$ for all $1 < j \leq u$ (see split transition construction in section 4.3 for the last two claims.) □

In the next definition we will denote by *valid markings* those markings in which the invariant relations associated with every control place hold and in which places obtained by transition split have the marking 0. Also we define equivalence of markings, which is an *equivalence relation* on the Petri nets $\mathcal{N}_1$, $\mathcal{N}_2$, $\mathcal{N}_3$, ... generated in each iteration. A class of equivalence contains the valid markings of the nets $\mathcal{N}_k$ which have the same marking for the places $p \in P_0$.

Definition 5.1 *Let $\mathcal{N}_i$, (L_i, b_i) and (L_{i0}, b_{i0}) be the Petri net and respectively the sets of constraints, all at the beginning of iteration $i \geq 1$, or for the initial Petri net, in which case $i = 0$. Let $\mathcal{C}$ be the set of control places that were added beginning with iteration 1 and $P_R = P_i \setminus (P_0 \cup \mathcal{C})$. A marking μ of $\mathcal{N}_i$ is said to be a* **valid marking** *if $\mu(p) = 0\ \forall p \in P_R$, $L_i\mu_e \geq b_i$ and $L_{i0}\mu_e \geq b_{i0}$, where μ_e is a marking of $\mathcal{N}_0$ such that $\mu_e(p) = \mu(p)\ \forall p \in P_0$, and the marking of the control places corresponds to the invariants they enforce.*

The definition above applies also for $\mathcal{N}_1$, where in case that no initial constraints exist, the remaining requirement for μ to be a valid marking of $\mathcal{N}_1$ is $\mu(p) = 0\ \forall p \in P_R$. When we refer to a marking μ of $\mathcal{N}_0$, μ is always valid when the algorithm starts with no constraints in (L_0, b_0). Otherwise, μ is valid if it satisfies the constraints stated at the beginning of the algorithm.

A Petri net $\mathcal{N}_i$ may not be *well-marked* for a marking that is valid. Indeed, the definition of valid markings does not require the *new* siphons of $\mathcal{N}_i$ not to be empty. Previous siphons cannot be empty for a valid marking, because of the constraints $L_i\mu_e \geq b_i$ and $L_{i0}\mu_e \geq b_{i0}$ which encode this requirement for previous siphons.

Definition 5.2 *Let μ_e be a* valid *marking of $\mathcal{N}_0$ and μ a* valid *marking of $\mathcal{N}_i$. If $\mu_e(p) = \mu(p)$ $\forall p \in P_0$, then μ_e and μ are said to be* **equivalent markings**. *Moreover, two* valid *markings μ_i of $\mathcal{N}_i$ and μ_j of $\mathcal{N}_j$ also are called* **equivalent markings** *if they have the same equivalent marking in $\mathcal{N}_0$.*

The way in which equivalence is defined implies that if two markings are equivalent they must also be valid. Equivalence is not defined for markings that are not valid.

Proposition 5.8 *Any valid marking of $\mathcal{N}_i$ has at most an equivalent marking in $\mathcal{N}_j$ for $0 \leq i < j$. Every valid marking of $\mathcal{N}_j$ has a unique equivalent marking in $\mathcal{N}_i$ when $0 \leq i < j$.*

Proof: By definitions 5.1 and 5.2, for any $\mathcal{N}_i$ a valid marking μ_i of $\mathcal{N}_i$ has a unique equivalent marking μ in $\mathcal{N}_0$. Also, μ_i is the unique equivalent marking of μ in $\mathcal{N}_i$. Indeed, the marking of the control places of $\mathcal{N}_i$ are the values of the excess variables associated to $L_i\mu \geq b_i$. The marking of the other places that do not appear in the original net $\mathcal{N}_0$ must be zero, in order that μ_i be valid. So μ_i can have only one equivalent marking μ_j in $\mathcal{N}_j$. The equivalent marking μ_j may not exist if μ, the equivalent marking of μ_i in $\mathcal{N}_0$, does not satisfy the additional constraints added in iterations $i, \ldots j-1$.

Because the constraints of iteration j, (L_j, b_j) and (L_{j0}, b_{j0}), include the constraints of iteration i, (L_i, b_i) and (L_{i0}, b_{i0}), it is clear that $L_j\mu \geq b_j \Rightarrow L_i\mu \geq b_i$ and $L_{j0}\mu \geq b_{j0} \Rightarrow L_{i0}\mu \geq b_{i0}$. So, if μ_j is a valid marking of $\mathcal{N}_j$, and μ_i is μ_j restricted to the places of $\mathcal{N}_i$, μ_i is also valid. By definition, if the marking μ of $\mathcal{N}_0$ is equivalent to μ_j then μ is μ_j restricted to the places of $\mathcal{N}_0$. Because μ_i and μ_j have the same equivalent marking in $\mathcal{N}_0$, they are therefore equivalent. □

Proposition 5.9 *The equivalence of markings is an equivalence relation.*

Proof: The proof follows immediately by checking the symmetry, reflexivity and transitivity of the relation. □

In [Moody, 1998] it was shown that adding control places to a net results in an incidence matrix of the form

$$D_2 = \begin{bmatrix} D_1 \\ D_c \end{bmatrix} \tag{16}$$

where D_1 is the incidence matrix of the initial net.

Proposition 5.10 *Let D_i and D_j be the incidence matrices of $\mathcal{N}_i$ and $\mathcal{N}_j$, $i < j$. If no transitions were split in iterations $i, \ldots j-1$, then D_j can be written in the form:*

$$D_j = \begin{bmatrix} D_i \\ D_c \end{bmatrix} \tag{17}$$

where the lines of D_c correspond to the control places added in iterations $i, \ldots j-1$.

Proof: Because no transitions were split, the inequalities enforced from iteration i to $j-1$ can be written only in term of the places of $\mathcal{N}_i$ (see section 4.5). Then, by enforcing these linear inequalities directly to $\mathcal{N}_i$ the closed loop is the same net as $\mathcal{N}_j$, and so the incidence matrix can be written as in equation (17) by Theorem 4.1 of section 4.1. □

The incidence matrix is more difficult to express when transitions are split, because some old columns disappear and new columns and rows appear.

Proposition 5.11 *Let μ_i and μ_k be two markings of $\mathcal{N}_i$ and $\mathcal{N}_k$, $i < k$.*

(a) μ_i and μ_k are equivalent markings if and only if they are valid and $\forall p \in P_i$, $\mu_i(p) = \mu_k(p)$.

(b) Assume that μ_i and μ_k are equivalent. Let t be an arbitrary transition of $\mathcal{N}_i$. If $\sigma_{i,k}(t)$ is enabled in $\mathcal{N}_k$, then t is enabled in $\mathcal{N}_i$. In addition, if $i \neq 0$, $t_{1,1}$ is the first transition of $\sigma_{i,k}(t)$ and $t_{1,1}$ is enabled, then t is enabled in $\mathcal{N}_i$.

(c) If S_i is a siphon of $\mathcal{N}_i^A$ and $\mu_k(p) = 0\ \forall p \in S_i$, then μ_k is not a valid marking of $\mathcal{N}_k$. However, if $\mu_i(p) = 0\ \forall p \in S_i$, μ_i may be a valid marking of $\mathcal{N}_i$.

(d) If μ_i is a valid marking and it does not have an equivalent marking in $\mathcal{N}_k$, j exists, such that $i \leq j < k$, $\mathcal{N}_j$ has a marking μ_j equivalent to μ_i and $\mathcal{N}_j^A$ has an empty siphon with respect to μ_j.

(e) If μ_i and μ_k are equivalent, $t \in T_0$, $\mu_i[\sigma_{0,i}(t) > \mu_i'$ and $\mu_k[\sigma_{0,k}(t) > \mu_k'$ then μ_i' and μ_k' are equivalent.

Proof: (a) Two markings are equivalent if they are valid. If valid, the marking of the places from replacement sequences are zero, while equivalence implies $\mu_i(p) = \mu_k(p)\ \forall p \in P_0$. The marking of the common control places of $\mathcal{N}_i$ and $\mathcal{N}_j$ are equal, being uniquely determined by the marking of the original places, for all valid markings (see section 4.5.) Hence the conclusion follows. On the other hand, by Proposition 5.8, μ_i and μ_j have equivalent markings $\mu_{0,i}$ and $\mu_{0,j}$ in $\mathcal{N}_0$. Because $P_0 \subseteq P_i$ and $\forall p \in P_i\ \mu_i(p) = \mu_k(p)$: $\mu_{0,i} = \mu_{0,j}$. Therefore μ_i and μ_j are equivalent.

(b) $\bullet t$ in $\mathcal{N}_i$ is subset or equal to $\bullet t_{1,1}$ in $\mathcal{N}_k$ ($\bullet t$ may be a subset because additional control places C with arcs $(C, t_{1,1})$ may appear in $\mathcal{N}_k$.) For $i \neq 0$, both $\mathcal{N}_i$ and $\mathcal{N}_k$ are PT-ordinary, and because μ_i and μ_k are equivalent, $t_{1,1}$ enabled implies t enabled. This may not be true for $i = 0$ because $\mathcal{N}_0$ may not be PT-ordinary. If $i = 0$, for all $p \in \bullet t$, firing $\sigma_{0,k}(t)$ requires that p has at least $W_0(p, t)$ tokens (see section 4.3.) Therefore t is enabled by μ_i.

(c) The deadlock prevention algorithm adds constraints for all uncontrolled siphons of the active subnet. So, the constraints (L_k, b_k) and (L_{k0}, b_{k0}) on $\mathcal{N}_k$ include the requirement that siphons of previous iterations be controlled. So μ_k cannot satisfy these constraints, and therefore is not a valid marking of $\mathcal{N}_k$. Further on, if S_i is not implicitly controlled by the constraints added in the iterations $1, 2, \ldots i-1$, there are valid markings of $\mathcal{N}_i$ such that S_i has no tokens.

(d) Let (L_x, b_x) and (L_{x0}, b_{x0}) be the constraints associated to $\mathcal{N}_x$, where $x > i$ is the first index such that μ_i does not satisfy one or both of (L_x, b_x) and (L_{x0}, b_{x0}). Because the requirements that are not satisfied only can correspond to the condition that some siphons of $\mathcal{N}_{x-1}^A$ be not empty, the conclusion follows for $j = x - 1$.

(e) Because no tokens remain in split replacement places by firing the entire sequences $\sigma_{0,i}(t)$ and $\sigma_{0,k}(t)$ replacing t, both μ_i' and μ_k' are valid. Let $\mu_{0,i}'$ and $\mu_{0,k}'$ be their equivalent markings in $\mathcal{N}_0$ and μ_0 the equivalent marking of μ_i and μ_k in $\mathcal{N}_0$. By part (b), $\mu_0[t > \mu_{0,i}'$ and $\mu_0[t > \mu_{0,k}'$. So $\mu_{0,i} = \mu_{0,k}$ and hence μ_i' and μ_k' are equivalent. □

Proposition 5.12 *Let $\mu_{i,1}$ and $\mu_{j,1}$ be two equivalent markings of $\mathcal{N}_i$ and $\mathcal{N}_j$, $i < j$. If $\mu_{i,2}$ and $\mu_{j,2}$ are two other equivalent markings of $\mathcal{N}_i$ and $\mathcal{N}_j$ and a transition t exists, such that $\mu_{i,1}[t > \mu_{i,2}$ in $\mathcal{N}_i$, then $\mu_{j,1}[\sigma_{i,j}(t) > \mu_{j,2}$ in $\mathcal{N}_j$.*

Proof: If $\sigma_{i,j}(t)$ is enabled by $\mu_{j,1}$ and $\mu_{j,1}[\sigma_{i,j}(t) > \mu'_{j,2}$ then $\mu'_{j,2}(p) = \mu_{i,2}(p)\ \forall p \in P_i$ and $\mu'_{j,2}(p) = 0\ \forall p \in P_s$ follow directly from split transition properties, where P_s is the set of the places resulted through transition split. Therefore, since $\mu_{j,1}$ is valid, $\mu'_{j,2}$ is also, because the constraints are satisfied (see Proposition 5.5). Then by Propositions 5.8 and 5.11(a), $\mu_{j,2} = \mu'_{j,2}$.

If $\sigma_{i,j}(t)$ is not enabled by $\mu_{j,1}$, let k be the first index such that $\sigma_{i,k}(t)$ is not enabled in $\mathcal{N}_k$ by $\mu_{k,1}$, which is the equivalent marking of $\mu_{i,1}$ in $\mathcal{N}_k$. Because $\sigma_{i,k-1}(t)$ is enabled in $\mathcal{N}_{k-1}$, there is a control place that prevents $\sigma_{i,k}(t)$ to fire, because of a constraint added in iteration $k-1$. So $\mu_{k-1,2}$ cannot satisfy one of the constraints added in iteration $k-1$, and therefore $\mu_{k-1,2}$ has no equivalent marking in $\mathcal{N}_k$. But this is a contradiction, because $j \geq k$ implies $\exists \mu_{k,2}$ equivalent to $\mu_{j,2}$ (Proposition 5.8), and $\mu_{j,2}$ is equivalent to $\mu_{i,2}$, which in turn is equivalent to $\mu_{k-1,2}$. (The fact that the markings $\mu_{i,2}$ and $\mu_{k-1,2}$ are equivalent follows from $\mu_{i,2}(p) = \mu_{k-1,2}(p)\ \forall p \in P_i$ because of the split transition construction (section 4.3), $\mu_{i,1}$ and $\mu_{k-1,1}$ are equivalent, $\mu_{i,1}[t > \mu_{i,2}$ in $\mathcal{N}_i$ and $\mu_{k-1,1}[t > \mu_{k-1,2}$ in $\mathcal{N}_{k-1}$.) □

Corollary 5.1 *Let $\mu^{(1)}$ and $\mu^{(2)}$ be two markings of $\mathcal{N}_0$ such that $\mu^{(1)}[t > \mu^{(2)}$ (where $t \in T_0$) and satisfying the constraints produced by the algorithm after termination: $L\mu^{(1)} \geq b$, $L_0\mu^{(1)} \geq b_0$, $L\mu^{(2)} \geq b$, $\mu^{(1)}[t > \mu^{(2)}$. Then the markings $\mu_k^{(1)}$ and $\mu_k^{(2)}$ of $\mathcal{N}_k$ equivalent to $\mu^{(1)}$ and respectively to $\mu^{(2)}$ are defined for any k, $\mu_k^{(1)}$ enables $\sigma_{0,k}(t)$ and $\mu_k^{(1)}[\sigma_{0,k}(t) > \mu_k^{(2)}$.*

Proof: Because $\mu^{(1)}$ and $\mu^{(2)}$ satisfy the constraints generated by the algorithm, all the control places that were added have a well defined marking, in accord with the supervisory policy. So $\mu_k^{(1)}$ and $\mu_k^{(2)}$ are defined for all iteration indices k. Then, by Proposition 5.12, the remainder of the conclusion follows. □

Theorem 5.1 *The following statements are true:*

(a) Let σ_i be an arbitrary firing sequence of $\mathcal{N}_i$ and $\sigma_j = \sigma_{i,j}(\sigma_i)$ the corresponding firing sequence in $\mathcal{N}_j$, $i < j$. If μ_j is a marking of $\mathcal{N}_j$ that enables σ_j, then the marking μ_i of $\mathcal{N}_i$ such that $\mu_i(p) = \mu_j(p)\ \forall p \in P_i$ enables σ_i. Also if $\mu_i[\sigma_i > \mu'_i$ and $\mu_j[\sigma_j > \mu'_j$ then $\mu'_i(p) = \mu'_j(p)$ $\forall p \in P_i$.

(b) Assume that the algorithm does not start with initial constraints, or if it does, all valid markings μ of $\mathcal{N}_0$ have the property that exists $\mu' \geq \mu$, μ' has an equivalent marking in $\mathcal{N}_k$. Let σ be an arbitrary transition sequence of $\mathcal{N}_0$ and $\sigma_k = \sigma_{0,k}(\sigma)$ the corresponding sequence in $\mathcal{N}_k$. If a valid marking μ of $\mathcal{N}_0$ exists which enables σ, a valid marking μ_k of $\mathcal{N}_k$ exists which enables σ_k.

(c) In the conditions of part (b), if some marking μ'_k of $\mathcal{N}_k$ exists which enables σ_k, then a marking of $\mathcal{N}_k$ exists which enables σ_k and which also is valid.

Proof: (a) If the property is true for all σ_i finite, than it is also true for all σ_i infinite. Indeed, if the property would not be true for some σ_i infinite, then there is a partition $\sigma_i = \sigma_{i,1}\sigma_{i,2}$ such that $\sigma_{i,1}$ is finite and $\sigma_{i,1}$ does not satisfy the property. Therefore, in what follows the proof considers only the case when σ_i is finite: $\sigma_i = t_1, t_2, \ldots t_s$, where every t_k is a transition of T_i.

The set P_j is the disjoint set union $P_j = P_i \cup \mathcal{C} \cup P_R$, where $\mathcal{C}$ is the set of control places added in the iterations i through $j-1$ and P_R is the set of places resulted from split transition operations in the same iterations. Firing $\sigma_{i,j}(t_1)$ requires the same number of tokens from places of P_i as firing t_1 in $\mathcal{N}_i$, as a split transition property, and may require additional tokens from $\mathcal{C}$. Therefore t_1 is enabled by μ_i. Let $\mu_{i,1}$ and $\mu_{j,1}$ be the markings reached by firing t_1 and $\sigma_{i,j}(t_1)$, respectively. Again, as a split transition property, firing t_1 in $\mathcal{N}_i$ and $\sigma_{i,j}(t_1)$ in $\mathcal{N}_j$ modifies in the same way the marking of P_i, and firing $\sigma_{i,j}(t_1)$ does not change the marking of P_R. Hence $\mu_{j,1}(p) = \mu_{i,1}(p)$ $\forall p \in P_i$ and $\mu_{j,1}(p) = \mu_j(p)$ $\forall p \in P_R$. Continuing in the same way with t_2, t_2 is enabled and the markings reached by firing t_2 and $\sigma_{i,j}(t_2)$ satisfy the same property, and by induction it follows that the markings $\mu_{i,1} \ldots \mu_{i,s}$ and $\mu_{j,1} \ldots \mu_{j,s}$ exist such that $\mu_i[t_1 > \mu_{i,1}[t_2 > \ldots \mu_{i,s-1}[t_s > \mu_{i,s}$, $\mu_j[\sigma_{i,j}(t_1) > \mu_{j,1}[\sigma_{i,j}(t_2) > \ldots \mu_{j,s-1}[\sigma_{i,j}(t_s) > \mu_{j,s}$, $\mu_{j,s}(p) = \mu_{i,s}(p)$ $\forall p \in P_i$ and $\mu_{j,s}(p) = \mu_j(p)$ $\forall p \in P_R$. So the conclusion follows with $\mu'_j = \mu_{j,s}$ and $\mu'_i = \mu_{i,s}$.

(b) This proof uses induction. Suppose that μ_i of $\mathcal{N}_i$ enables the sequence q. Let $\mathcal{S}_0$ denote the set of siphons of $\mathcal{N}_i^A$ which in $\mathcal{N}_i$ either are token-free under the marking μ_i, or become so by firing q. By Proposition 5.3(c) each siphon $s \in \mathcal{S}_0$ includes at least an original place and/or a control place. Using the relations from section 4.5, a valid marking $\mu_{i,2} \geq \mu_i$ can be chosen such that $\forall s \in \mathcal{S}_0$, $\sum_{p \in s} \mu_{i,2}(p) \geq \sum_{p \in s} \mu_i(p) + 1$. By construction, for the marking $\mu_{i,2}$ no siphon s is token-free, $\mu_{i,2}$ also enables q and no siphon s becomes token-free when firing q. Thus $\mu_{i,2}$ has an equivalent marking μ_{i+1} which enables q in $\mathcal{N}_{i+1}$.

(c) Let P_R be the set of all places of $\mathcal{N}_k$ that have resulted through transition split in previous iterations. Let μ''_k be defined as $\mu''_k(p) = \mu'_k(p)$ $\forall p \in P_k \setminus P_R$ and $\mu''_k(p) = 0$ $\forall p \in P_R$. Then μ''_k enables σ_k. Indeed, let's assume the contrary. Then σ_k can be partitioned in the sequence $\sigma_k = \sigma_1 t_x \sigma_2$, where $t_x \in T_k$, $\mu''_k[\sigma_1 > \mu'_x$, $\mu'_k[\sigma_1 > \mu_x$, μ_x enables t_x but μ'_x does not enable t_x. The only possibility is that $P_R \cap {}^\bullet t_x = \{p_x\}$, $\mu_x(p_x) > 0$ and $\mu'_x(p_x) = 0$ (refer also to the split transition construction in section 4.3.) Because $\sigma_k = \sigma_{0,k}(\sigma)$ and σ is a sequence of transitions of $\mathcal{N}_0$, σ_1 has the form $\sigma_{0,k}(t_1)\sigma_{0,k}(t_2) \ldots \sigma_{0,k}(t_n)\sigma_x$, where $t_1, \ldots t_n$ are not necessarily distinct transitions of T_0 and σ_x is the first part of some $\sigma_{0,k}(t_{n+1})$. It follows that $\sigma_{0,k}(t_{n+1})$ has the form $\sigma_x t_x \sigma_y$. However, firing σ_x always brings a token in the replacement place p_x such that $p_x{}^\bullet = \{t_x\}$, which contradicts $\mu'_k(p_x) = 0$.

Because μ''_k enables σ_k, we can always choose a valid marking μ_k such that $\mu_k \geq \mu''_k$ (see the form of the constraints added by the algorithm in section 4.5.) Therefore μ_k is valid and enables σ_k. □

Corollary 5.2 *Consider the assumption of Theorem 5.1(b) to be true.*

(a) Deadlock-freedom cannot be enforced for any finite marking in $\mathcal{N}_k$ if and only if it also cannot be enforced in $\mathcal{N}_0$.

(b) Liveness cannot be enforced for any finite marking in $\mathcal{N}_k$ if and only if it also cannot be enforced in $\mathcal{N}_0$.

Proof: Deadlock-freedom may be enforced in a net in which there is a marking allowing an infinite firing sequence. Thus necessity results directly from Theorem 5.1(b) and sufficiency from Theorem 5.1 parts (a) and (c), where part (c) is used for the case when initial constraints exists, and so not all possible markings of $\mathcal{N}_0$ are valid. The proof of part (b) is similar. □

Theorem 5.1(a) showed that if $i < j$ and μ_i, μ_j are equivalent markings of $\mathcal{N}_i$ and $\mathcal{N}_j$, then a firing sequence σ_i is always enabled by μ_i in $\mathcal{N}_i$, when its counterpart $\sigma_j = \sigma_{i,j}(\sigma)$ is enabled by μ_j in $\mathcal{N}_j$. The converse generally is not true. However, it is true for the particular case when $i = 0$, because $\mathcal{N}_1$ differs from $\mathcal{N}_0$ only by the fact that $\mathcal{N}_1$ is the PT-transformed version of $\mathcal{N}_0$ and no constraints were yet enforced.

Proposition 5.13 *Every valid marking μ of $\mathcal{N}_0$ has an equivalent marking μ' in $\mathcal{N}_1$. Moreover, if μ and μ' are equivalent, σ is a transition sequence enabled by μ and $\sigma' = \sigma_{0,1}(\sigma)$, then μ' enables σ'.*

Proof: The equivalent marking μ' of μ is defined by $\mu'(p) = \mu(p)\ \forall p \in P_0$ and $\mu'(p) = 0\ \forall p \in P_1 \setminus P_0$. The fact that $\forall t \in T_0$, $\mu[t > \mu_1$ implies both $\mu'[\sigma_{0,1}(t) > \mu'_1$ and μ_1 is equivalent to μ'_1, is a property of transition split. Thus the remainder of the conclusion follows immediately. □

5.2 Main Results

The first important result of this section is Theorem 5.2, that shows when the algorithm the algorithm provides a supervisor preventing deadlock and when the algorithm detects that no supervisor preventing deadlock exists. Theorem 5.3 gives a practical way to guarantee termination. Theorem 5.4 gives a permissivity estimate of the supervisor generated by the algorithm: the supervisor is at least as permissive as any supervisor enforcing liveness, if any exists.

This section uses the same notations as in the description of the algorithm in section 4.7, as well as the notations from section 5.1.1. That is, in every iteration i the *inactive subnet* is $\mathcal{N}_i^I = (P_i^I, T_i^I, F_i^I, W_i^I)$, the *active subnet* $\mathcal{N}_i^A = (P_i^A, T_i^A, F_i^A, W_i^A)$ and the *total net* $\mathcal{N}_i = (P_i, T_i, F_i, W_i)$, $\sigma_{i,j}(\sigma)$ the replacement sequence in $\mathcal{N}_j$ of the transition sequence σ of $\mathcal{N}_i$, $i < j$ and $\sigma_{i,j}(t)$ the replacement sequence in $\mathcal{N}_j$ of the transition t of $\mathcal{N}_i$.

The following lemma shows that the final active subnet is deadlock-free when no minimal siphon exists which has a transition or split replacement taking, when fired, more than one token from the siphon without returning any back. This condition of Lemma 5.1 will be referred to guarantee that the algorithm provides a supervisor preventing deadlock.

Lemma 5.1 *Assume that the algorithm terminates after $k-1$ iterations and that in the final active subnet $\mathcal{N}_k^A$ no minimal siphons S exist such that $\exists t_x \in T_0$, $\sigma_{0,k}(t_x) = t_{x,1}t_{x,2}\ldots t_{x,u}$, $t_{x,u} \notin {}^\bullet S$,*

$F_x = \{(p,t) : p \in S, t \in \sigma_{0,k}(t_x), (p,t) \in F_k^A\}$ *and* F_x *has at least two elements.* $(\mathcal{N}_k^A, \mu^A)$ *is deadlock-free for all markings* μ^A *that are the restriction of a valid marking of* $\mathcal{N}_k$ *to the places of* $\mathcal{N}_k^A$.

Proof: Minimal siphons of the active subnet in a certain iteration may be found transformed in subsequent iterations, as shown in Proposition 5.4. The modification is that they may include additional places resulted through transition split operations. A minimal siphon is controlled by adding a control place if the control place is not in the situation of Proposition 5.6(b-ii), which appears when there are transitions taking more than one token from the siphon without returning any back. This property corresponds to the siphon having the property forbidden in the statement of the lemma. The siphons not having the forbidden property remain controlled (for valid markings) in the subsequent iterations, if they still appear in the active subnet (see Proposition 5.7.) However it is also true that any siphon with the forbidden property becomes controlled if in subsequent iterations it loses all transitions or replacement sequences satisfying the forbidden property (by being moved to the inactive subnet), since their only mean of becoming empty disappears (see proof of Proposition 5.6(b)). Because the algorithm terminated, all minimal siphons have been considered for control. Since none of the minimal siphons has the forbidden property, all are controlled and by Proposition 3.1 the subnet is deadlock-free. □

Recall that if $\mathcal{N}_k$ is a PT-ordinary Petri net with no uncontrolled siphons, Proposition 3.1 suggests that we found an supervisor enforcing deadlock-freedom for $\mathcal{N}_0$. However, when the original net is not repetitive, i.e. liveness cannot be enforced, the final net may have a nonempty inactive subnet. So the final *active subnet* is not equal to the final *total net*. Therefore it is not yet clear that if the final *active subnet* is PT-ordinary and has no uncontrolled siphons, then the *total net* is deadlock-free. The next theorem proves among other things that this is the case.

Theorem 5.2 *Assume that the* deadlock prevention method *terminates after* $k-1$ *iterations. Let* $\mathcal{N}_0$ *be the original Petri net and* $\mathcal{N}_k$ *the net produced by the last iteration. Let* (L, b) *and* (L_0, b_0) *denote the two sets of constraints generated by the algorithm.*

(a) Any transition t *of* $\mathcal{N}_0$*, such that* $t_x \in T_k^I$ *and* t_x *appears in* $\sigma_{0,k}(t)$*, cannot be made live for any finite marking of* $\mathcal{N}_0$.

(b) If $\mathcal{N}_k^A$ *is an empty net (i.e.* $P_k^A = \emptyset$*,* $T_k^A = \emptyset$*), then the original net* $\mathcal{N}_0$ *cannot be made deadlock-free.*

(c) If the conditions of Lemma 5.1 hold true and $\mathcal{N}_0$ *cannot be made deadlock-free, then either (i) or (ii) is true:*

(i) $\mathcal{N}_k^A$ *is an empty net.*

(ii) the algorithm started with initial constraints (see also Theorem 5.3) and there is no marking of $\mathcal{N}_0$ *which satisfies all constraints.*

(d) If $\mathcal{N}_k^A$ *is nonempty and the conditions of Lemma 5.1 hold true, then the original net* $\mathcal{N}_0$ *in closed loop with the supervisor enforcing* $L\mu \geq b$*, is deadlock-free for all initial markings* μ_0 *of* $\mathcal{N}_0$*, such that* $L\mu_0 \geq b$ *and* $L_0\mu_0 \geq b_0$.

Proof: (a) The algorithm updates the subnets in step 3 by repeated passing through the two steps of section 4.6. A transition introduced in the *inactive subnet* is not later split. So if such a transition t cannot be made live in $\mathcal{N}_i$, it cannot be made live either in $\mathcal{N}_j$, $j > i$, by Theorem 5.1(a).

The proof is by induction. Let t_f be the first transition introduced in the *inactive subnet*, and let i be the iteration number when this happens. By the algorithm construction, $\exists p \in P_i$ s.t. $p \in \bullet t_f$ and $\bullet p = \emptyset$. Therefore t_f cannot fire infinitely often in $\mathcal{N}_i$. Suppose that at a certain point all transitions in the *inactive subnet* have the property that they cannot be made live. Let t be the next transition introduced in the *inactive subnet* and let j be the iteration number when it happens. A transition t is copied in the *inactive subnet* when $\exists p \in P_j$, s.t. $p \in \bullet t$ and $\bullet p$ is a subset of the current set of transitions of the *inactive subnet*. So, t cannot fire infinitely often in $\mathcal{N}_j$ because in p only can enter finitely many tokens. This proved that all transitions in T_k^I cannot be made live. Then the conclusion follows by Theorem 5.1(b). Indeed, if the algorithm had no initial constraints, t live in $\mathcal{N}_0$ for some marking implies t_x live in $\mathcal{N}_k$ for some marking, by Theorem 5.1(b), which contradicts $t_x \in T_k^I$. If t cannot be live when there are no marking constraints, it cannot be live either with marking constraints. So the proof covers the case when the algorithm starts with initial constraints as well.

(b) Obvious from (a), since no transition can be made live.

(c) The proof is by contradiction. Assume that $\mathcal{N}_k^A$ is nonempty and that there is an initial marking of $\mathcal{N}_0$ which satisfies the constraints. By part (d) (whose proof follows) this is not possible.

(d) By construction, every marking of the original Petri net $\mathcal{N}_0$ which satisfies the constraints has an equivalent marking in $\mathcal{N}_k$ such that $\mathcal{N}_k^A$ is well-marked. The proof uses the fact that for any such marking, there is an infinite sequence enabled in $\mathcal{N}_k^A$ (Lemma 5.1). It proves by contradiction that no marking of $\mathcal{N}_0$ satisfying the constraints is a deadlock marking for the closed loop Petri net.

Assume that from a good initial marking μ_0 of $\mathcal{N}_0$, the closed loop net (let it be $\mathcal{N}_S$) reaches a marking μ such that all possible firings in $\mathcal{N}_0$ would lead to markings which do not comply with the enforced constraints, $L\mu \geq b$. This would be deadlock in $\mathcal{N}_S$.

Let $\mu_{0,k}$ and μ_k be the equivalent markings of μ_0 and μ in $\mathcal{N}_k$, and μ_k^A the restriction of μ_k to the places of $\mathcal{N}_k^A$. Because μ_k is valid, by Lemma 5.1 μ_k^A enables an infinite transition sequence σ in $\mathcal{N}_k^A$. Let T_R be the set of transitions that appeared by split transition operations and $T_f \subseteq T_R$ the set of transitions which are last in the sequences of split transition replacements. Let $\mathcal{C}$ be the set of control places. Revisiting the transition split operation (section 4.3) and by Proposition 5.2(b), firing any $t \in T_R \setminus T_f$ always reduces the marking of some places in $P_0 \cup \mathcal{C}$ and firing $t \in T_f$ increases the marking of some places in $P_0 \cup \mathcal{C}$. Because the total marking of $P_0 \cup \mathcal{C}$ is finite, σ must include transitions $t \in T_0 \cup T_f$ (where from T_0 may only appear transitions that remained unsplit). Let t_1 be the first transition in $T_0 \cup T_f$ that appears in σ. If $t_1 \in T_0$, since all transition of σ before t_1 are in $T_R \setminus T_f$, and firing them only decrease markings of $P_0 \cup \mathcal{C}$, t_1 is enabled by μ_k^A since it is enabled after firing the transitions that precede it in σ. But this implies that t_1 is also enabled by μ in $\mathcal{N}_S$, which is a contradiction. The remaining possibility is $t_1 \in T_f$, and so t_1 is the last transition $t_{x,m}$ of a split replacement sequence $\sigma_1 = t_{x,1} \dots t_{x,m}$. Since μ_k^A is valid, $t_{x,m-1}$

must appear in σ before t_1, $t_{x,m-2}$ must appear in σ before $t_{x,m-1}$, and so on. Because all other transitions of σ not in σ_1 that appear before t_1 are in $T_R \setminus T_f$, μ_k^A enables σ_1. This is contradiction, because this implies that μ enables $t_0 \in T_0$ such that $\sigma_1 = \sigma_{0,k}(t_0)$. □

Perhaps the most important result of the previous theorem is part (d), which gives conditions that the method provides a supervisor preventing deadlock. Part (a) of Theorem 5.2 motivates step 3 of the algorithm. Indeed, the algorithm removes from the active subnet all transitions connected to source places. These transitions cannot fire infinitely often, and after becoming inactive (dead), the Petri net behaves as if they were missing from the net. Removing them from the active subnet is useful to reveal new siphon structures.

Corollary 5.3 *If at the end of some iteration $k-1$ the* total net $\mathcal{N}_k$ *has a source place, liveness is not enforcible in $\mathcal{N}_0$.*

Proof: Let p be a place such that $\bullet p = \emptyset$ and $p\bullet \neq \emptyset$. By the construction of the deadlock prevention algorithm, $p\bullet \subseteq T_k^I$. Then, by theorem 5.2 (a) $\mathcal{N}_0$ cannot be made live. □

A Petri net $\mathcal{N}$ is **structurally bounded** [Murata] if for all finite markings μ_0, $\mathcal{R}(\mathcal{N}, \mu_0)$ is bounded. The algorithm can be guaranteed to terminate for such Petri nets if step 2 of the algorithm considers only new siphons that are not *implicitly controlled* (see section 4.5.) A sufficient condition that Theorem 5.2(d) applies for this modification of the algorithm is that the final active subnet contains no replacements of split transitions, because in this case it is clear that the implicitly controlled siphons of the active subnet are controlled siphons and so the proof of Lemma 5.1 is unchanged.

Theorem 5.3 *Let $\mathcal{N}$ be a structurally bounded Petri net. Let $\mathcal{M}_I$ be a set that includes all possible initial markings in some given application. If $\mathcal{M}_I$ is bounded, then a supervisor based on the method of section 4.7 can be constructed in a finite number of iterations.*

Proof: Consider modifying the step 2 of the algorithm to consider only new siphons that are not *implicitly controlled* (section 4.5.) Since $\mathcal{N}$ is structurally bounded and $\mathcal{M}_I$ is bounded, the set of reachable markings is bounded. Let $\mathcal{M}_R$ be a bounded set that includes the set of reachable markings. Let F_N be the set of markings forbidden by the control places added up to some point. Let S be the next siphon considered for control, and f_S the set of markings which would be forbidden in the original net by enforcing $\sum_{p \in S} \mu(p) \geq 1$ (i.e. by adding a control place). S is not implicitly controlled if $f_S \setminus F_N \neq \emptyset$. Not every marking might be reached, so the previous condition can be written as $(f_S \setminus F_N) \cap \mathcal{M}_R \neq \emptyset$. Since each controlled siphon adds at least a new forbidden marking that is in $\mathcal{M}_R$, and $\mathcal{M}_R$ is finite, after we control a finite number of siphons, all new siphons are implicitly controlled. □

Considering the assumptions of Theorem 5.3 true, let (L_i, b_i) be a set of linear constraints defining a bounded feasible set that includes the set of reachable markings $\mathcal{R}(\mathcal{M}_I)$. Then the

deadlock prevention algorithm can be started with initial constraints (L_0, b_0) equal to (L_i, b_i), and by Theorem 5.3 it terminates. Theorem 5.3 is important because it gives a sufficient (but not necessary) condition for termination which is not very restrictive for real applications, where in general the capacity of every node is finite.

Lemma 5.2 *Consider the case when $\mathcal{N}_0$ is repetitive. Let S be a siphon of $\mathcal{N}_{i+1}$, $i \geq 1$, that does not appear in $\mathcal{N}_i$. Let μ_{i+1} be a valid marking of $\mathcal{N}_{i+1}$ and μ_i the equivalent marking in $\mathcal{N}_i$. Assume that S is empty. Let t_s be an arbitrary transition of $\mathcal{N}_i$ with the property that there is a transition $t \in S\bullet$ of $\mathcal{N}_{i+1}$ such that $t_s = t$ or t_s is split in $\mathcal{N}_{i+1}$ and t appears in the replacing transition sequence $\sigma_{i,i+1}(t_s)$. If $\exists \mu \in \mathcal{R}(\mu_i)$ such that $\mu[t_s > \mu_s$, then $(\mathcal{N}_i, \mu_s)$ has at least one empty siphon.*

Proof: Let $\mathcal{C}$ be the set of control places added to $\mathcal{N}_{i+1}$. Note that P_{i+1} is made up of P_i, $\mathcal{C}$ and the set of places that result through transition split, $P_R = P_{i+1} \setminus (P_i \cup \mathcal{C})$. Let σ be the firing sequence that was used to reach μ: $\mu_i[\sigma > \mu$. We consider the parallel evolution of $\mathcal{N}_i$ and $\mathcal{N}_{i+1}$ from the equivalent markings μ_i and μ_{i+1}, by firing the transitions of σ in $\mathcal{N}_i$ and the same transitions or their replacements in $\mathcal{N}_{i+1}$. The only reason for $\sigma' = \sigma_{i,i+1}(\sigma)$ not to be enabled in $\mathcal{N}_{i+1}$ by μ_{i+1} would be that a control place prevents it.

If σ' is not enabled, $\sigma = \sigma_1 t_1 \sigma_2$, $\mu_i[\sigma_1 > \mu_1$, $\mu_{i+1}[\sigma_{i,i+1}(\sigma_1) > \mu_1'$, μ_1 enables t_1, but μ_1' does not enable $\sigma_{i,i+1}(t_1)$. This corresponds to the following: $\mathcal{N}_i$ has a siphon S_1, that is controlled in $\mathcal{N}_{i+1}$ with C_1; when C_1 was added, $t_1 \in C_1\bullet$, and if $W(C_1, t_1) > 1$, t_1 was split in step 3 of iteration i in $\sigma_{i,i+1}(t_1)$ or if $W(C_1, t_1) = 1$, $\sigma_{i,i+1}(t_1) = t_1$. So $t_1 \in S_1\bullet$, and since t_1 would not be allowed by C_1 to fire from μ_1, it means that firing it would make S_1 empty. Since t_1 is fired in the sequence $\sigma = \sigma_1 t_1 \sigma_2$, after σ is fired, S_1 is an empty siphon in $(\mathcal{N}_i, \mu_s)$.

If σ' is enabled by μ_{i+1}, let μ' be the marking reached: $\mu_{i+1}[\sigma' > \mu'$. Because σ' may contain only entire replacements of split transitions and μ_{i+1} is a valid marking (which implies $\mu_{i+1}(p) = 0$ $\forall p \in P_R$), $\mu'(p) = 0$ $\forall p \in P_R$. Also, μ_{i+1} and μ_i are equivalent and $\sigma' = \sigma_{i,i+1}(\sigma)$, therefore $\mu(p) = \mu'(p)$ $\forall p \in P_i$ (Theorem 5.1(a)). Because S is a siphon, S empty for μ_{i+1} implies S empty for all reachable markings, and so for μ' too. There are two cases: (a) t_s is not split in $\mathcal{N}_{i+1}$ and (b) t_s is split.

(a) If t_s is not split, $\bullet t_s \cap P_R = \emptyset$. Further on, μ enables t_s in $\mathcal{N}_i$ but μ' does not enable t_s in $\mathcal{N}_{i+1}$, so in $\mathcal{N}_{i+1}$, $\bullet t_s \cap \mathcal{C} \neq \emptyset$ and there is $C \in \bullet t_s \cap \mathcal{C}$ such that $\mu'(C) = 0$. Let S_C be the siphon of $\mathcal{N}_i$ controlled by C. t_s was not split, so $W(C, t_s)$ was 1; t_s enabled by μ, $\mu'(C) = 0$ and $t_s \in C\bullet$ $\Rightarrow t_s \in (S_C\bullet) \setminus (\bullet S_C)$. S_C appears in $\mathcal{N}_{i+1}$ either unmodified or as a siphon $S_C' \subseteq S_C \cup P_R$. Since $\mu'(p) = 0$ $\forall p \in P_R$, $S_C \subseteq P_i$ and $\mu'(C) = 0$, $\sum_{p \in S_C} \mu(p) = 1$, where Proposition 5.5 was also applied for μ' seen as reachable from the valid marking μ_{i+1}. Because t_s is enabled by μ, firing t_s empties S_C, so there is an empty siphon in $(\mathcal{N}_i, \mu_s)$.

(b) If t_s was split, then let $\sigma_{i,i+1}(t_s) = t_{s,1} \ldots t_{s,r}$ be its replacement in $\mathcal{N}_{i+1}$. Also, let $p_{s,1} \ldots p_{s,r-1}$ be the places that appeared through the split; the notations follow the convention from section 4.3, i.e. $t_{s,i} = \bullet p_{s,i}$, $i = 1 \ldots r-1$, etc. From the lemma statement, $\exists p \in S \cap \bullet t_{s,u}$. If $u > 1$, we prove that $p = p_{s,u-1}$.

If $p \neq p_{s,u-1}$, then $p \notin P_R$, because by construction of split transitions, every place p_s produced by split has $| \bullet p_s| = |p_s \bullet | = 1$. Also, $p \notin P_i$, because (see section 4.3) this implies in $\mathcal{N}_i$ that $W_i(p_i, t_s) > 1$, which is not possible since the Petri net at the beginning of every iteration $i \geq 1$ is PT-ordinary. The remaining possibility $p \in \mathcal{C}$ also is not possible. Indeed, this implies that there is a control place C which, after being added to $\mathcal{N}_i$, produced an arc (C, t_s) with $W(C, t_s) > 1$. But $C \in S$, S is empty, so $\mu'(C) = 0$, which implies that S_C, the siphon of $\mathcal{N}_i$ controlled by C, has only one token at the marking μ, so μ will not enable t_s since t_s needs more than one token from S_C to fire. Contradiction.

So $p = p_{s,u-1}$. Because $p \in S$ and $\bullet p_{s,u-1} = t_{s,u-1}$, $t_{s,u-1} \in S\bullet$. This showed that $t_{s,u} \in S\bullet \Rightarrow t_{s,u-1} \in S\bullet$. So, $t_{s,1} \in S\bullet$. Since t_s is enabled by μ, and S is empty for μ', $\bullet t_{s,1} \cap S \subseteq \mathcal{C}$. As before, there is a control place $C \in S \cap (\bullet t_{s,1})$ controlling a siphon S_C of $\mathcal{N}_i$; $C \in \bullet t_{s,1}$ implies $C \in \bullet t_s$ before t_s was split, and so firing t_s reduces the marking of S_C. Because this is 1 at marking μ, firing t_s makes S_c empty. Therefore there is an empty siphon in $(\mathcal{N}_i, \mu_s)$. □

Part (a) of Theorem 5.2 shows that if liveness may be enforced in the original net $\mathcal{N}_0$, then the *inactive subnet* remains empty, and the final supervised net is equal to the final *active subnet*. Theorem 5.4 addresses this case, in which for every k, $\mathcal{N}_k^A = \mathcal{N}_k$.

Theorem 5.4 *The deadlock prevention method provides a supervisor at least as permissive as any liveness enforcing supervisor, if any.*

Proof: Assuming that there are markings which allow a liveness enforcing policy, $\mathcal{N}_0$ is repetitive, by Corollary 3.1(b), and Lemma 5.2 applies. The proof is by contradiction. It shows that any marking forbidden by the deadlock prevention method also is forbidden by any liveness enforcing supervisor. Recall that our algorithm forbids markings which will produce an empty siphon in an $\mathcal{N}_k$ for some k.

Let $\mu^{(1)}$ be a marking of $\mathcal{N}_0$ and $\mu_k^{(1)}$ the equivalent marking in $\mathcal{N}_k$. Suppose that for the marking $\mu_k^{(1)}$ there is an empty siphon S_k in $\mathcal{N}_k$. Because $\mu_k^{(1)}$ is valid, S_k is a new siphon which does not appear in $\mathcal{N}_{k-1}$; $\mu^{(1)}$ is forbidden by iteration k, which adds the constraint that S_k be well-marked.

Assume that $\mu^{(1)}$ is not forbidden by some liveness enforcing supervisor and that there is an infinite firing sequence σ enabled by $\mu^{(1)}$ such that every transition of $\mathcal{N}_0$ appears infinitely often in σ. According to Lemma 5.2, there is a transition t'_{k-1} of $\mathcal{N}_{k-1}$ such that in any possible firing sequence, after t'_{k-1} fires in $\mathcal{N}_{k-1}$, there is an empty siphon S_{k-1} of $\mathcal{N}_{k-1}$. Let $t_{k-1} \in T_0$ such that t'_{k-1} appears in $\sigma_{0,k-1}(t_{k-1})$. Let $\mu^{(2)}$ be the marking of $\mathcal{N}_0$ that appears while σ is fired, immediately after t_{k-1} fires for the first time. Also, let σ_1 be the subsequence of σ that was fired so far, that is $\mu^{(1)}[\sigma_1 > \mu^{(2)}$. Let $i \geq 0$ be the largest integer, such that $\mu_i^{(2)}$ is an equivalent marking of $\mu^{(2)}$ in $\mathcal{N}_i$. By Lemma 5.2, $i \leq k-1$. Indeed, if σ_1 is allowed to fire in $\mathcal{N}_{k-1}$, there is an empty siphon S_{k-1} for the marking $\mu_{k-1}^{(2)}$, but there is no valid marking of $\mathcal{N}_k$ such that S_{k-1} is empty (Proposition 5.11(c)). Now, the fact that $\mu^{(2)}$ has an equivalent marking $\mu_i^{(2)}$ in $\mathcal{N}_i$ but not in $\mathcal{N}_{i+1}$ shows that there is an empty siphon S_i in $\mathcal{N}_i$ and that S_i does not appear in $\mathcal{N}_{i-1}$ (Proposition

5.11(d)). Further on, the same idea as before is used, that a transition t_{i-1} with the same property as t_{k-1} exists, and following the same idea, an index $j \leq i-1$ is found such that for the marking $\mu^{(3)}$ of $\mathcal{N}_0$ there is an empty siphon in $\mathcal{N}_{j-1}$. This procedure is repeated and finally two cases may appear (Lemma 5.2 applies for $i > 0$ only) after the first n transitions of σ are fired, where n is a finite number. Let σ_p denote the sequence that enumerates the first n transitions of σ, and let $\mu^{(p)}$ be the marking reached by firing σ_p (that is, $\mu^{(1)}[\sigma_p > \mu^{(p)}$) and $\mu_1^{(p)}$ the equivalent marking in $\mathcal{N}_1$. Then (a) there is an empty siphon in $(\mathcal{N}_0, \mu^{(p)})$ or (b) there is an empty siphon in $(\mathcal{N}_1, \mu_1^{(p)})$. Case (a) contradicts the fact that every transition appears infinitely often in σ and $\mu^{(1)}$ enables σ, since after n firings none of the transitions in the postset of the empty siphon may fire again. Case (b) leads to the same type of contradiction, because Proposition 5.13 shows that the firing sequence $\sigma' = \sigma_{0,1}(\sigma)$ is enabled by $\mu_1^{(1)}$, which is the equivalent marking of $\mu^{(1)}$ in $\mathcal{N}_1$, and by construction every transition of $\mathcal{N}_1$ appears infinitely often in σ'. □

In other words, the theorem states that the set of forbidden markings of the supervisor obtained by successive application of the siphon controlling rule is a subset of the set of markings forbidden by any liveness enforcing supervisor. The previous result shows also that if for some Petri net the successive application of the supervisory controlling rule enforces liveness, the resulting supervisor is maximally permissive. The theorem applies for a supervisor obtained after an arbitrary number of iterations. The proof does not assume that the algorithm terminates.

5.3 Special Cases

5.3.1 Additional Constraints

We consider the case when additional constraints are to be enforced. Let (L_a, b_a) be the additional constraints and $\mathcal{N}$ the Petri net. A good way to proceed with the deadlock prevention algorithm is to apply it rather to the supervised Petri net $\mathcal{N}_L$, which contains the additional places necessary to enforce (L_a, b_a) according to the invariant based approach ([Moody, 1998], also outlined in section 4.1). So the algorithm would start with $\mathcal{N}_0 = \mathcal{N}_L$ and initial constraints (L_0, b_0) reflecting (L_a, b_a).

The reason why it is *not* a good idea to apply the deadlock prevention algorithm first to $\mathcal{N}$ and then to enforce (L_a, b_a) is that additional constraints can make deadlock possible. Indeed, we can easily find examples of deadlock-free Petri nets which with additional marking constraints may reach deadlock.

5.3.2 Finite Capacity Petri Nets

In many applications it is reasonable to assume that the maximum number of tokens that a place may have is bounded. In this case the Petri nets may be extended with an additional function K which maps its capacity to each place. This type of Petri net is called place/transition net [Reisig]. So, a **place/transition structure** is represented by the quintuple $\mathcal{N} = (P, T, F, W, K)$, where $K : P \to \overline{\mathbb{N}}$ is the **capacity function**, and with an additional initial marking we have a

place/transition net, denoted by $(\mathcal{N}, \mu_0)$. The capacity of a place is allowed to be infinite. The firing rule of a transition in place/transition nets is the same as for conventional Petri nets, except that a transition is not enabled by a marking if firing it would cause a place to exceed its capacity.

Let $\mathcal{N} = (P, T, F, W, K)$ be a place/transition structure and $\mathcal{N}_R = (P, T, F, W)$ the corresponding Petri net structure. $\mathcal{N}$ can be transformed in an equivalent conventional Petri net $\mathcal{N}_E$ by enforcing in $\mathcal{N}_R$, to each place p with finite capacity, the linear constraint $\mu(p) \leq K(p)$. The conventional Petri net is obtained using the invariant based approach of [Moody, 1998], outlined also in section 4.1.

If all the places have finite capacity, the equivalent Petri net is by construction structurally bounded. The deadlock prevention algorithm can be started as in section 5.3.1, with $\mathcal{N}_0 = \mathcal{N}_E$ and constraints (L_a, b_a) which describe $\mu(p) \leq K(p)$ for all $p \in P$. The method can be guaranteed to terminate by Theorem 5.3, since a bound on the marking of each place is known. Indeed, the upper bound for the marking of any place $p \in P$ is the finite capacity $K(p)$ and the upper bound for the marking of a control place p_c enforcing for a place $p \in P$ the constraint $\mu(p) \leq K(p)$, is also $K(p)$.

5.3.3 Safe Petri Nets

An ordinary Petri net $(\mathcal{N}, \mu_0)$ is **safe** if for all reachable markings the marking of any place is at most 1. We consider the case when a Petri net $\mathcal{N}$ needs to be made safe by supervision. The deadlock prevention algorithm may be used to provide such a policy which is not blocking.

Let (L_a, b_a) be the constraints associated to $\mu(p) \leq 1$, for all places of $\mathcal{N}$. Then we can proceed as shown in section 5.3.1.

The deadlock prevention algorithm terminates by Theorem 5.3, because it is known that 1 is an upper bound of the marking of each place.

5.3.4 Some Particular Cases when Liveness is also Enforced

It is possible, however not very likely, that if the initial Petri net is an asymmetric choice net the final Petri net still will be an asymmetric choice net. By Theorem 3.1, this is a sufficient condition for liveness for all initial markings which are not forbidden.

Both parts of Corollary 3.2(c) are useful for the deadlock prevention algorithm. The second part is good also because of Theorem 5.4. Corollary 3.2(c) provides conditions that let us know before applying the algorithm whether the supervisor also will enforce liveness. In the case of asymmetric choice net result, we need first to run the algorithm, and then check whether the final result complies with Theorem 3.1.

It is not clear at this time if the conditions of Corollary 3.2(c) have practical importance. It depends on whether or not there is an efficient algorithm to check them.

The class of Petri nets on which the algorithm enforces liveness may be larger then that resulting

from Corollary 3.2(c), because the class of deadlock prevention supervisors more permissive than liveness enforcing supervisors is rather large.

Note that whenever the supervisor provided by the algorithm enforces liveness, it is the maximally permissive supervisor, by Theorem 5.4.

5.4 Final Remarks and Directions for Further Research

5.4.1 The Termination Problem

Theorem 5.3 shows how we can guarantee the termination of the algorithm in the case of structurally bounded Petri nets. The termination of the algorithm is facilitated by considering only minimal siphons that are not *implicitly controlled* (see section 4.5). For instance, the algorithm does not terminate for the Petri net of figure 9 if implicitly controlled siphons are not eliminated. However this operation does not guarantee termination in general. For instance, if in figure 9 we change the weight of (t_2, p_1) to 2, the algorithm does not terminate, failing to generate one of the good constraints. Instead it generates a sequence of constraints converging to that constraint. When $W(t_2, p_1) = 1$ that good constraint is generated from a siphon appearing in iteration 2, which does not appear for $W(t_2, p_1) = 2$, and which allows to consider as controlled the siphon that generates the recurrent behavior.

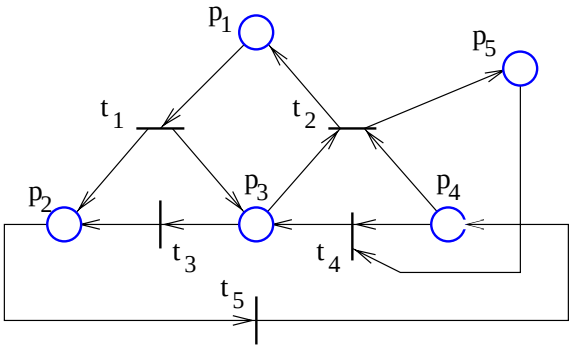


Figure 9: Example for the termination problem

Checking whether a siphon is implicitly controlled is equivalent to an integer programming feasibility problem, which is an *NP* type problem [Wolsey].

5.4.2 An Improvement that is Desired for Permissivity

5.4.2.1 A Desired Permissivity Property It has been proven that the method provides a supervisor at least as permissive as any liveness enforcing supervisor, if any. It would be desirable to modify the method such that we have a similar permissivity result for Petri nets which cannot

be made live. In this case, the requirement would be that the supervisor is at least as permissive as any supervisor that insures that any transition that can be made live is live for any reachable marking. (This makes sense, since it has been proven in Corollary 3.3 that the transitions of any Petri net may be divided in transitions which eventually become dead, for any finite marking, and transitions that can be made live).

An illustration of this problem is in example 4.2. One may check that the marking vector $[0, 0, 2]$, which allows enforcement of the discussed permissivity property, is unacceptable for the constraints (L, b) which were generated by the deadlock prevention algorithm.

5.4.2.2 A Possible Solution Progress has been made in this direction based on the idea that a minimal siphon may be allowed to include source places if we do not consider them as siphons. This idea can be generalized as follows (probably there are more ways to do it).

Let P_D be the set of places which are not live for any marking, in the sense that eventually (after some firings, depending on the initial marking) they cannot enable any transitions. Let P_A be the rest of places, i.e. places connected to transitions that can be made live. Places may be divided in the disjoint sets P_D and P_A with a polynomial computational complexity (see appendix).

An *active siphon* is a siphon of the total net that contains all places of some siphon of the *active subnet* (where the active subnet is the total net obtained by deleting all transitions that cannot be made live and all places remained thus unconnected). An active siphon is minimal if it contains no other active siphon. The algorithm would successively control all minimal active siphons (instead of minimal siphons, as it is done in the algorithm of this paper). It can be shown in the case when no transitions need to be split (and probably for the more general case too) that this modified algorithm prevents deadlock and has the maximally permissive property defined above. A drawback is that if there are siphons in the active subnet such that deadlock can be avoided even for markings in which they are empty, this algorithm might allow them to become empty. So this algorithm variant is rather closer to deadlock prevention than liveness enforcement.

5.4.2.3 A Harder Permissivity Problem As it was said in previous parts of the work, the algorithm does not provide in general the maximum permissive deadlock prevention supervisor. This limitation appears because of the principle on which the method is based, that a deadlocked PT-ordinary Petri net has an empty siphon (Proposition 3.1), is only a necessary condition.

On the other hand, it is not clear if in practice it really would be useful to allow a system to reach some local deadlocks, and only to prevent reaching a state of total deadlock. Basically, the method of this work prevents siphons from becoming empty. An empty siphon is a form of local deadlock, since all transitions connected to the empty siphon can no longer fire.

5.4.3 Other Remarks

A MATLAB implementation revealed the fact that a key factor for the applicability of the deadlock prevention method is a fast program for (minimal) siphon computation. As the number of iterations

increases, the size of the Petri net increases with new control places, split transitions and places from replacements of split transitions. Both computation of siphons and checking whether new siphons are implicitly controlled have a computation time which grows rapidly with the number of places and constraints, respectively. For this reason it seems to be faster in general to reduce the redundancy of the constraints at every iteration, and so to remove at every iteration that control places which become redundant because of stronger new constraints.

6 Summary of Results

This paper introduced a deadlock prevention algorithm for Petri nets. The algorithm was stated in section 4.7. The main results concerning the deadlock prevention algorithm were proved in section 5.2. They show that:

- When the algorithm terminates given a Petri net $\mathcal{N}_0$ and the requirement of Lemma 5.1 applies, then $\mathcal{N}_0$ in closed loop with the supervisor enforcing the constraints (L, b) is deadlock-free for all initial markings μ_0 such that $L_0\mu_0 \geq b_0$ and $L\mu_0 \geq b$ (Theorem 5.2(d)).
- The case when the structure of $\mathcal{N}_0$ does not allow deadlock to be prevented for any finite initial marking is also detected when the algorithm terminates and the requirement of Lemma 5.1 is true. The conditions are given in Theorem 5.2(b) and (c).
- The algorithm does not necessarily generate the maximally permissive supervisor which prevents deadlock. However it has the permissivity quality that if there are initial markings μ_0 such that liveness can be enforced in $(\mathcal{N}_0, \mu_0)$, the supervisor provided by the algorithm (for deadlock-freedom) is at least as permissive as any supervisor enforcing liveness (Theorem 5.4).
- There are particular cases in which the supervisor of our algorithm also enforces liveness. In these cases the algorithm provides the maximally permissive liveness enforcing supervisor.
- The algorithm can be slightly modified to guarantee termination for structurally bounded Petri nets, by assuming an upper bound of the marking of each place to be known; see proof of Theorem 5.3. This assumption is reasonable in problems that require initial markings from a bounded set. A sufficient condition which guarantees deadlock prevention for this case is that the final active subnet contains no split transition replacements.

The properties of section 5.1.2 are technical results important for understanding the method and for proving the main results. In section 3.2, several consequences are derived from a known result [Murata], which appears in this paper as Theorem 3.2. These consequences are general results that give insight in the relation of deadlock prevention to liveness enforcement. Part (b) of Corollary 3.1 also appears in [Sreenivas, 1997]. Of particular interest for this paper are Corollary 3.3 and Corollary 3.2.

A major advantage of our approach is that it is very general, being applicable to generalized Petri net structures. The approach is also applicable for timed Petri nets and with some modifications

it is potentially applicable to Petri nets with uncontrollable and unobservable transitions by using the admissible constraint transformation from [Moody, 1998]. Another advantage of the algorithm is that it does not require the initial marking to be known and that it characterizes the usable initial markings as the feasible region of a set of linear inequalities.

APPENDIX

A Computation of Transitions that Cannot Be Made Live

Below we show an algorithm of polynomial complexity that identifies the set T_D from Corollary 3.3. If the transitions from T_D are removed, dividing the resulting net in active and inactive subnets will identify the set of places P_D that are connected only to transitions which cannot be made live: $P_D = P \setminus P_A$, where P is the set of places and P_A the set of places in the active subnet.

A.1 Problem Formulation

Let x be a vector and A a matrix. Let $\|x\|$ denote the support of x: $\|x\| = \{i : x(i) \neq 0\}$ and let $\mathcal{M}(A) = \{y : Ay \geq 0, y \geq 0\}$. We say that $x \in S$ has *maximum support* with respect to a set S if $\forall y \in S$, $\|y\| \subseteq \|x\|$. It is not difficult to check that $\exists x \in \mathcal{M}(A)$ that has maximum support.

A.2 A Linear Programming Approach

In this approach, to see whether an index i belongs to the maximum support, a linear program is solved. The program checks a solution exists for the constraints $Ax \geq 0$, $x(i) = 1$ and $x \geq 0$. With artificial variables, $Ax \geq 0$ is transformed in $[A, -I] \cdot [x^T, y^T]^T = 0$.

It may not be clear from the beginning why we solve linear programs and not linear integer programs. Obviously, for any rational invariant q, an integer invariant x exists with the same support; indeed, by multiplying q with the denominators of the nonzero elements of q we get such an invariant. A less obvious result is that for any real invariant r, a rational invariant exists with the same support. This is not true in general (for instance consider the invariants of $A = [1, \pi]$), but it is true for matrices with rational elements.

Let $|x - y|$ be the Euclidean norm of $x - y$.

Proposition 1.1 *Let A be a matrix with rational elements. Let x be a real vector in the null space of A. For all $\epsilon > 0$ a rational vector q exists such that $|x - q| < \epsilon$ and q has the same support as x.*

Proof: Let A_r be A restricted to the columns i of A such that $x(i) \neq 0$, and let x_r be the

corresponding restriction of x. For simplicity, assume that A_r is full row rank (otherwise we can delete a number of rows) and that the first $rank(A_r)$ columns of A_r are linearly independent (otherwise we can reorder the columns). Then $A_r = [B, N]$, and if $x = [x_B^T, x_N^T]^T$, we have that $x_B = -B^{-1}Nx_N$. All $x_N(i)$ that are irrational can be approximated as close as desired with rational numbers; so let q_N be a rational vector such that $|q_N - x_N| < \delta$. Then $q_B = B^{-1}Nq_N$ is rational, because A has rational elements, and $|q_B - x_B| \leq |B^{-1}N||q_N - x_N|$. Therefore, if $\delta = \epsilon/(\sqrt{1 + |B^{-1}N|^2})$, then $|q_r - x_r| < \epsilon$, and by adding 0 elements we find a rational vector q such that $|q - x| < \epsilon$. To make sure that q and x have the same support, δ can be chosen small enough such that for $x_r(i) > 0$: $q_r(i) > 0$ and for $x_r(i) < 0$: $q_r(i) < 0$. □

The proposition above shows that computation of integer invariant supports can be accomplished by working with real invariants. So Integer Programming is not necessary in this application, and the more efficient methods of Linear Programming can be used instead.

The algorithm is outlined below:

- Transform $Ax \geq 0$ in $[A, -I] \cdot [x^T, y^T]^T = 0$, where y are the excess variables. Let n be the number of rows of x.
- Is index i found to belong to a nonnegative invariant? If yes, try index $i + 1$. If not, continue with the next step.
- If $i > n$, terminate.
- Check feasibility of $[A, -I] \cdot [x^T, y^T]^T = 0$, $x(i) = 1$, $x \geq 0$ and $y \geq 0$ with a Linear Programming method. If feasible, let $[x_s^T, y_s^T]^T$ be a solution. Add all indices in $\|x_s\|$ to the set of indices that belong to a nonnegative invariant.

This method is more efficient than the method based on invariant computation because it has a good computational complexity. The worst case is better than $O(n * LP)$, where LP corresponds to the order of the complexity of the Linear Programming method that is used. When an interior point method is used, the algorithm always has a polynomial complexity.

References

[1] Barkaoui, K., I. Abdallah, (1995) "Deadlock Avoidance in FMS Based on Structural Theory of Petri Nets," *IEEE Symposium on Emerging Technologies and Factory Automation*, 1995.

[2] Barkaoui, K., J.-F. Pradat-Peyre, (1996) "On Liveness and Controlled Siphons in Petri Nets," in *Application and Theory of Petri nets*, Springer Verlag, 1996.

[3] Boer E, T. Murata, (1994) "Generating Basis Siphons and Traps of Petri Nets Usign the Sign Incidence Matrix," *IEEE Transactions on Circuits and Systems*, 41(4).

[4] Coffman E., Elphick M., Shaoshani A., (1971) "System Deadlocks," *Computing Surveys*, vol. 3, pp.67-68, June 1971.

[5] Commoner F., (1972) *Deadlocks in Petri nets*, Applied Data Research Inc., Wakefield, Massachusetts 01880, Report Nr. CA-7206-2311, 1972.

[6] David R., A. Hassane, (1994) "Petri Nets for Modeling of Dynamic Systems – A Survey," in *Automatica*, vol. 32, No. 2, 1994.

[7] Dijkstra E., (1965) "Cooperating Sequential Processes," in *Programming Languages*, Genuys F. editor, London, Academic Press, 1965.

[8] Ezpeleta J., J. Couvreur, M. Silva, (1993), "A New Technique for Finding a Generating Family of Siphons, Traps and ST-Components. Application to Colored Petri Nets," in *Advances in Petri Nets*, *Lecture Notes in Computer Science*, Springer-Verlag 1993.

[9] Ezpeleta J., J. Colom, J. Martinez, (1995) "A Petri Net Based Deadlock Prevention Policy for Flexible Manufacturing Systems," *IEEE Trans. on Robotics and Automation*, Vol. 11, no. 2, April 1995.

[10] Giua A., F. DiCesare, M. Silva, (1992) "Generalized Mutual exclusion Constraints on Nets with Uncontrollable Transitions," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 974-979, Chicago, October 1992.

[11] Giua A., F. DiCesare, (1994) "Blocking and Controllability of Petri Nets in Supervisory Control," *IEEE Transactions on Automatic Control*, vol. 39, no. 4, pp. 818-823, April 1994.

[12] Hack M., (1972) *Analysis of Production Schemata by Petri Nets*, Technical Report 94, Project MAC, 1972.

[13] Ichikawa A., Hiraishi K., (1987) "Analysis and Control of Discrete Event Systems Represented by Petri Nets," in *Discrete Event Systems: Models and Applications*, IIASA Conference, Sopron, Hungary 1987, Springer Verlag 1988.

[14] Iordache M., (1999) *Deadlock Prevention in Discrete Event Systems Using Petri Nets*, Master's Thesis, University of Notre Dame.

[15] Krogh B., (1987) "Controlled Petri Nets and Maximally Permissive Feedback Logic," in *Proceedings of 25th Annual Allerton Conference*, University of Illinois, Urbana, 1987.

[16] Lautenbach K., (1987) "Linear Algebraic Calculation of Deadlocks and Traps," in *Concurrency and Nets*, Springer-Verlag 1987.

[17] Lautenbach K., H. Ridder, (1994) "Liveness in Bounded Petri Nets which are Covered by T-Invariants," in *Applications and Theory of Petri Nets*, *Lecture Notes in Computer Science*, p. 358-375, Springer-Verlag 1994.

[18] Lautenbach K., H. Ridder, (1996) "The Linear Algebra of Deadlock Avoidance — A Petri Net Approach," Research Report at Institute for Computer Science, University of Koblenz, Germany, 1996.

[19] Lewis F., H. Huang, D. Tacconi, A. Gürel, O. Pastravanu, (1998) "Analysis of Deadlocks and Circular Waits Using a Matrix Model for Discrete Event Systems," *Automatica*, vol. 34, no. 9, 1998.

[20] Moody J., K. Yamalidou, M. Lemmon, P. Antsaklis, (1994) "Feedback Control of Petri Nets Based on Place Invariants," in *Proceedings of the 33rd IEEE Conference on Decision and Control*, volume 3, pp. 3104-3109, Lake Buena Vista, December 1994.

[21] Moody, J. P. Antsaklis, (1996) "Supervisory Control of Petri Nets with Uncontrollable/Unobservable Transitions," in *Proceedings of the 35th IEEE Conference on Decision and Control*, pp. 4433-4438, Kobe, Japan, December 1996.

[22] Moody, J., P. Antsaklis, (1998) *Supervisory Control of Discrete Event Systems Using Petri Nets*, Kluwer Academic Publishers.

[23] Moody, J., P. Antsaklis, (1999) "Petri Net Supervisors for DES with Uncontrollable and Unobservable Transitions" to appear in *IEEE Transactions on Automatic Control*, 1999.

[24] Murata, T. (1989) "Petri Nets: Properties, Analysis and Applications," in *Proceedings of the IEEE*, Vol. 77, No. 4, April 1989.

[25] Nash S., A. Sopher, (1996) *Linear and Nonlinear Programming*, McGraw-Hill Companies Inc.

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

[27] Sinha P., (1996) *Distributed Operating Systems*, IEEE Press, 1996.

[28] Sreenivas R., (1997) "On the Existence of Supervisory Policies that Enforce Liveness in Discrete Event Systems Modeled by Controlled Petri Nets," in *IEEE Transactions on Automatic Control*, Vol. 42, No. 7, July, 1997.

[29] Sreenivas R., (1998) "An Application of Independent, Increasing, Free-Choice Petri Nets to the Synthesis of Policies that Enforce Liveness in Arbitrary Petri Nets," in *Automatica*, Vol. 34, No. 12, December, 1998, pp. 1613-1615.

[30] Sreenivas R., (1999) "On Supervisory Policies that Enforce Liveness in in a Class of Completely Controlled Petri Nets obtained via Refinement," in *IEEE Transactions on Automatic Control*, Vol. 44, No. 1, January, 1999.

[31] Tanenbaum A., (1987) *Operating Systems*, Prentice-Hall.

[32] Walukiewicz S., (1991) *Integer Programming*, Kluwer Academic Publishers.

[33] Wolsey L., (1998) *Integer Programming*, New York: John Wiley & Sons.

[34] Yamalidou K., J. Moody, M. Lemmon, P. Antsaklis, (1994) "Feedback control of Petri nets based on place invariants," Technical Report of the ISIS Group ISIS-94-002.2, University of Notre Dame, May 1994.

[35] Yamalidou K., J. Moody, M. Lemmon, P. Antsaklis, (1996) "Feedback control of Petri nets based on place invariants," in *Automatica*, vol. 32, no.1, January 1996.