Deadlock Control of Automated Manufacturing Systems Based on Petri Nets - A Literature Review

ZhiWu Li
School of Electro-Mechanical Engineering
Xidian University
No.2 South Taibai Road
Xi’an 710071, China
email: systemscontrol@gmail.com

NaiQi Wu, Senior Member, IEEE
Department of Mechatronics Engineering
Guangdong University of Technology
Guangzhou 510090, China
e-mail: nqwu@gdut.edu.cn

MengChu Zhou
Department of Electrical and Computer Engineering
New Jersey Institute of Technology
Newark, NJ 07102, USA
e-mail: zhou@njit.edu

Contact Information:
Office Location: A1111, New Sci-Tech Building, North-Campus, Xidian University
Access ID: SCATR201008002
Tel: 86-29-88-20-24-41
Fax: 86-29-88-20-43-38
Website: http://sca.xidian.edu.cn/

2010.09.03
Deadlock Control of Automated Manufacturing Systems Based on Petri Nets - A Literature Review

ZhiWu Li, Senior Member, IEEE
School of Electro-Mechanical Engineering
Xi’an JiaoTong University
No.2 South TaiBai Road
Xi’an 710071, China

Institute of Computer Science
Martin Luther Universität of Halle-Wittenburg
Kurt-Mothes Straße, 1
06120 Halle, Germany
email: zhiwu.li@informatik.uni-halle.de

NaiQi Wu, Senior Member, IEEE
Department of Mechatronics Engineering
Guangdong University of Technology
Guangzhou 510090, China
e-mail: nqwu@gdut.edu.cn

MengChu Zhou, Fellow, IEEE
Department of Electrical and Computer Engineering
New Jersey Institute of Technology
Newark, NJ 07102, USA
e-mail: zhou@njit.edu

Abstract

Deadlocks are a rather undesirable situation in a highly automated flexible manufacturing system. Their occurrences often deteriorate the utilization of resources and may lead to catastrophic results in safety-critical systems. Graph theory, automata, and Petri nets are three important mathematical tools to handle deadlock problems in resource allocation systems. Particularly, Petri nets are considered as a popular formalism due to its inherent characteristics. They have received much attention over the past two decades to deal with deadlock problems in automated manufacturing systems, leading to a variety of deadlock control policies. This research surveys the state-of-the-art deadlock control strategies for automated manufacturing systems by reviewing the principles and techniques involved in preventing, avoiding, and detecting deadlocks. The primary focus of this research is deadlock prevention due to a large and continuing stream of efforts on it. A control strategy is evaluated in terms of computational complexity, behavioral permissiveness, and structural complexity of its liveness-enforcing or deadlock-free supervisor. This comprehensive study provides readers a conglomeration of the updated results in this area and facilitates engineers in finding a suitable approach for their industrial application cases. Future research directions are finally discussed.

Index Terms: Petri net, deadlock prevention, deadlock avoidance, discrete event system, flexible manufacturing systems

1 Introduction

Popularized by Henry Ford in the early of the 20th century, mass production greatly contributes to the development and progress of human society and civilization. However, over the last three decades, it was challenged by quick market changes of multiple product
types with a small batch. The survivability of a manufacturing system to a large extent depends on its capability to swiftly respond to the variable market requirements. Such a technological edge can be achieved through state-of-the-art manufacturing technologies, leading to the emergence and development of flexible manufacturing systems (FMS) [14], [54]. An FMS is a conglomeration of computer numerically controlled machine tools, buffers, fixtures, robots, automated guided vehicles (AGV), and other material-handling devices. It usually exhibits a high degree of resource sharing in order to increase flexibility such that market changes can be responded quickly. The existence of resource sharing may lead to circular wait conditions, the real cause of deadlocks in which each of a set of two or more jobs keeps waiting indefinitely for the other jobs in the set to relinquish resources that they hold. In such a system, once deadlocks occur, they persist and would not be resolved without the intervention from human beings or other external agency.

With the wide and extensive applications of information technology to contemporary manufacturing systems, their safety, reliability, and some other miscellaneous complicated requirements are met by control software [32]. Actually, modern automated manufacturing systems are increasingly becoming software-intensive systems. Compared with their establishment and maintenance, the design of control software is a complex, intricate, and arduous task for control engineers by whom deadlock problems must be carefully considered and appropriately handled since deadlocks can lead to the stoppage of a part or the whole of a system, but also give rise to catastrophic results in highly automated systems such as semiconductor manufacturing and critical distributed databases. Deadlock problems in automated manufacturing systems have received more and more attention from both academic and industrial communities.

The deadlock issues are originated from resource allocation systems, which can be traced as far back as to the memory assignments in an operating system by computer scientists in 1960’s [41], [83], [84], [87], [106], [161], [166], [191]. As a logical problem, deadlocks can arise in different contexts. Although a variety of deadlock control methods are developed for deadlock problems in operating systems, multiprocessing, and distributed databases, they cannot directly be applied to manufacturing systems due to different technical background. For example, as resources in a computer operating system, memory is preemptable and can be taken away from the process owning it. While resources in a manufacturing system are usually non-preemptable. Deadlocks in a resource allocation system are in general considered to be a result of (1) deficiency of system resources, (2) an inappropriate order of process execution, and (3) improper resource allocation logic. Summarily, there are four necessary conditions for a deadlock to occur, known as the Coffman conditions first proposed in [41]:

1. Mutual exclusion condition: a resource that cannot be used by more than one process at a time;
2. Hold and wait condition: processes already holding resources may request new resources;
3. No preemption condition: no resource can be forcibly removed from a process holding it, and resources can be released only by the explicit action of the process; and
4. Circular wait condition: two or more processes form a circular chain where each process waits for a resource that the next process in the chain holds.

It is shown that the first three conditions are decided by the physical characteristics of a system and its resources. That is to say, for a system with a given set of resources, the first three conditions are either true or false. They are time-invariant. However, the fourth
can be enforced to vary depending on request, allocation, and release of the resources in the system. Once a deadlock occurs, all the four conditions must hold. On the contrary, a deadlock will never occur if one of these conditions is not satisfied.

Breaking the mutual exclusion condition implies that no process can exclusively access a resource. This mechanism is impossible for a resource that cannot be spooled. Deadlocks may still exist even if all the resources in a system can be spooled. A manufacturing resource, e.g., a machine tool and a robot, cannot usually be spooled.

Removing the hold and wait condition can be achieved by requiring a process to request and be allocated all resources that are needed before its execution. Alternatively, a process is allowed to request resources only when it does not hold any resources. The former mechanism is too conservative and often leads to unacceptable low resource utilization, while the latter is generally impractical in manufacturing systems.

The preemptability of a resource depends on its nature. If a process that is holding some resources requests another resource that cannot be immediately allocated to it, then all the resources currently being held are released. However, this mechanism is usually unrealistic, difficult, or even impossible in manufacturing practice.

Circular wait prevention allows processes to wait for resources but ensures that the waiting is not circular. The common and popular circular wait avoidance mechanism is to establish off-line a precedence to each resource and force processes to request resources in order of increasing precedence. This forces resource allocation to follow a particular and non-circular ordering. Hence, circular wait cannot occur.

Before the emergence of agile and flexible manufacturing, the occurrences of deadlocks in a system are infrequent since most manufacturing resources are non-shared. Once a deadlock occurs, it can be solved by the intervention of operators through aborting one or more processes involved in the deadlock. Such a strategy sounds brute-force but can lead to high system throughput since deadlocks are usually local and do not result in a global stoppage of a system. An FMS, however, aims to provide a trade-off solution between productivity of mass production and flexibility for multiple part types. Resource sharing is its a inherent and elementary feature. The occurrence of a deadlock degrades system performance. Deadlock-freedom is also a key requirement for safety-critical systems. For example, in a highly automated semiconductor manufacturing system, the amount of time that wafers stay at a chamber is absolutely crucial. The occurrence of a deadlock extends the sojourn of wafers at chambers, possibly making all the wafers discarded in the chambers of the processes involved in the deadlock. As a result, it is essential to ensure that deadlock will never occur.

The necessity of the four conditions for a deadlock to occur leads us to infer that negating one of them makes impossible the occurrence of deadlocks in a resource allocation system. The physical characteristics and technical background of an automated flexible manufacturing system show that the first three deadlock conditions always hold and the only feasible doorway to eliminate deadlocks is to falsify the circular wait condition [71], [?].

Digraphs, automata, and Petri nets are three major mathematical tools to investigate deadlock problems in FMS [71]. Graph theory or a digraph is a simple and intuitive tool to describe interactions between operations and resources, from which a deadlock control policy can be derived. The representative research group are led by Wysk [38], [121], [231], [232], and Fanti [62], [64], [65], [66], [67], [68], [69], [70]. Supervisory control theory (SCT) [179] based on formal languages and finite automata, originated by Ramadge and Wonham, aims at providing a comprehensive and structural treatment of the modeling and control of discrete event systems. As an important paradigm, SCT has a profound influence on the supervisory control of automated manufacturing systems under other for-
malisms such as Petri nets. A number of effective yet computationally efficient deadlock control policies are developed based on automata. Lawley, Reveliotis, and Ferreira are distinguished experts in this area [125], [126], [127], [128], [129], [130], [131], [132], [133], [183], [236]. In particular, a theoretically significant deadlock avoidance policy with polynomial-complexity is developed for a class of resource allocation systems in [184], which is then described in a Petri net formalism [171].

The publications on deadlock control of FMS indicate that Petri nets are increasingly becoming an important, popular, and fully-fledged mathematical model to investigate the modeling and control of FMS. Figure 1(a), 1(b), and 1(c) show the numbers of publications in Engineering Index database [55] over the past three decades on deadlock problems in resource allocation systems by using digraphs, automata, and Petri nets as formalisms, respectively. The publications indicate that more and more studies employ Petri nets as a tool to handle deadlock problems. This study hence focuses on the literature review of deadlock control approaches based on a Petri net formalism.

The rest of paper is organized as follows. Section 2 outlines deadlock-handling strategies existing in the literature, whose merits and drawbacks are briefly discussed in an FMS context. Approaches to deadlock detection and recovery, deadlock avoidance, and deadlock prevention are reviewed in Sections 3, 4, and 5, respectively. Section 6 proposes open problems in this area. Section 7 concludes this paper and Section 8 presents bibliography notes.

2 Deadlock-handling Strategies

From a conceptual standpoint, there are four strategies to handle deadlocks in automated manufacturing systems: deadlock-ignoring, deadlock prevention, deadlock avoidance, and deadlock detection and recovery.

Deadlock-ignoring, known as the Ostrich algorithm [91], is employed in a resource allocation system if the probability of deadlocks is tiny and the enforcement of other deadlock control strategies is technically or financially difficult. A notable example of deadlock-ignoring is in UNIX where a deadlock can occur if the process table is full and all the processes still attempt to fork more subprocesses. This situation, however, appears rather infrequently and to prevent this scenario needs cumbersome restrictions, which is usually ignored. In theory, deadlock-ignoring strategy is a trade-off between convenience and correctness. In an automated manufacturing system, deadlock-ignoring is feasible and reasonable from the technical and economic points of view if the degree of resource-sharing is not high. This strategy is widely used in the early development stage of flexible manufacturing systems.

Deadlock prevention is considered to be a well-defined problem in resource allocation studies. It is usually achieved by using an off-line computational mechanism to control the request for resources to ensure that deadlocks never occur. In other words, resources are granted to requesting processes in such a way that a request for a resource never leads to a deadlock. The goal of a deadlock prevention approach is to impose constraints on a system to prevent it from reaching deadlock states. In this case, the computation is carried out off-line in a static way and once a control policy is established, the system can no longer reach undesirable deadlock states. The simplest way to prevent a deadlock is to acquire all the needed resources before a process starts its execution. This is highly inefficient since it decreases system concurrency and stifles its operational flexibility. A major advantage of deadlock prevention algorithms is that they require no run-time cost since problems are solved in system design and planning stages. The major criticism is that they tend to be too conservative, thereby reducing the resource utilization and system
Figure 1: Engineering Index: (a) digraphs, (b) automata, (c) Petri nets.
productivity. Deadlock prevention is usually considered to be applicable to safety-critical systems in which deadlocks may lead to serious results and cause enormous economic loss.

In a deadlock avoidance strategy, a resource is granted to a process only if the resulting state is safe. A state is called safe if there exists at least one execution sequence that allows all processes to run to completion. In order to decide whether the forthcoming state is safe if a resource is allocated to a process, every cell controller and the global controller need to keep track of the global system state. This means that huge storage and extensive communication ability are necessary. This is also true for the case that an automated manufacturing system operates under control of a distributed control system in which there is no the global controller. Furthermore, it is computationally expensive to check the safety of a state due to a large number of reachable states in a real-world system. It is worthy noting that too aggressive methods usually lead to higher resource utilization and throughput, but do not totally eliminate all deadlocks for some cases. In such cases if a deadlock arises, suitable recovery strategies are still required [121], [213], [232]. Conservative methods eliminate all unsafe states and deadlocks, and often some good states, thereby degrading the system performance. On the other hand, they are intended to be easy to implement.

In deadlock detection and recovery, however, resources are granted to a process without any check. The status of resource allocation and requests are examined periodically to determine whether a set of processes is deadlocked. This examination is performed by a deadlock detection algorithm. If a deadlock is found, the system recovers from it by aborting one or more deadlocked processes. A primary requirement for a correct deadlock detection algorithm is that it must detect all possible deadlocks and does not report nonexistent deadlocks. In manufacturing practice, human operators are often needed for this strategy and thus can be very expensive.

The persistence of a deadlock means the stoppage of a part or the whole system. Deadlock recovery is by no means a trivial task. In fact, the problem of promptly and efficiently resolving a detected deadlock is as important as the deadlock detection itself.

A deadlock is resolved by aborting one or more processes involved in the deadlock and granting the released resources to other processes involved in the deadlock. A process is aborted by withdrawing all its resource requests, restoring its current state to an appropriate previous state, relinquishing all the resources that it has acquired after that state, and restoring all the relinquished resources to their original states. In the simplest case, a process is aborted by starting it afresh and relinquishing all the resources that it holds [191]. As stated previously, a deadlock is resolved by aborting at least one or more process involved in the deadlock and granting the released resources to other processes involved in the deadlock. Usually, a deadlock resolution involves the following steps that are computationally expensive.

1. Select a victim, a process to be aborted, for the optimal resolution of a deadlock.
2. Abort the victim, release all the resources held by it, restore all the released resources to their previous states, and grant the released resources to deadlocked processes.

In a flexible manufacturing system, if the cost of aborting a process involved in a deadlock is high, technically difficult, or impossible, a feasible alternative is the usage of spare resources that can function as those involved in deadlocked processes. This strategy however implies extra investment.

The suitability of a deadlock-handling strategy to a large extent depends on the application cases. Both deadlock prevention and deadlock avoidance are conservative. They are usually used if the deadlocks are frequent or if the occurrence of a deadlock leads to a
serious result. In contrast, deadlock detection and recovery are an optimistic strategy that grants a resource to a request as long as the resource is available. Hopefully, this resource allocation does not lead to a deadlock. As a result, deadlock detection and recovery are suitable for the system in which deadlocks are rare, their occurrences do not lead to severe results, and their recovery is technically or financially affordable.

3 Deadlock Detection and Recovery

The stability of deadlocks is shown by their persistence. Once a deadlock occurs, it persists forever. As a result, it is not difficult to detect deadlocks. For example, the existence of a cycle in the resource allocation graph of a system is a necessary condition for deadlocks. An algorithm to detect a cycle in a graph requires an order of \( n^2 \) operations, where \( n \) is the number of vertices in the graph, i.e., the number of resources in general manufacturing practice.

Deadlocks in an automated manufacturing system can be detected by a prior simulation through executing the most favored processing sequences as long as possible in order to determine the most probable processes that are involved in a deadlock as well as the resources held by these permanently blocked processes. In theory, such an approach, just as if one stands by a tree stump waiting for a hare, cannot completely disclose all deadlock scenarios in a systems. Formal tools to detect deadlocks include resource allocation graphs, Petri nets, and other paradigms and methods. For example, a well-known result is the Deadlock Theorem: \( S \) is a deadlock state if the resource allocation graph of \( S \) is not completely reducible [87]. Moreover, it is shown that the existence of a cycle in a resource allocation graph is a necessary condition for deadlocks. In some special case, it is also sufficient. In a Petri net formalism, deadlocks are closely tied to a structural object, namely siphons. Once a siphon in the Petri net modeling a system is unmarked (i.e., no tokens sojourn in it), all its related transitions, i.e., events, are dead, implying the permanent blocking of some or all processes in the system.

If a system is in a deadlock state, some recovery strategies to activate it must be applied so that the deadlock is resolved. Three kinds of actions can be taken for recovery: process termination, resource preemption, and resource reservation.

1. Process termination: Aborting all deadlocked processes is usually a bad bargain. Alternatively, one can kill a process at a time until the deadlock disappears, which, however, implies high overhead since a deadlock detection algorithm is run at each step.

2. Resource preemption: A victim, i.e., a process that is not one involved directly in the deadlock, is carefully selected from which enough resources are preempted such that they are made available to deadlocked processes to resolve the deadlock. Thereafter, the process whose resources are preempted has to be rolled back to some previous safe state and is made to continue its operations. However, resource preemption is in general infeasible in flexible manufacturing systems.

3. Resource reservation: As a similar case of the resource preemption strategy, this deadlock resolution approach calls resources reserved beforehand for the deadlocked processes such that the deadlock can be eliminated.

In an automated FMS, it is hard to imagine that the three mentioned strategies can be implemented automatically. That is to say, a deadlock in a manufacturing system is
in general resolved through manual intervention whatever strategies are employed, which also implies high cost.

Deadlock recovery can be considered as a special case of error recovery that has been extensively investigated over the past years [167], [168], [180], [213], [214], [247]. A graph-theoretical approach is introduced in [63] for deadlock detection and recovery in FMS whose resources have multiple capacities. The concept of maximal-weight zero-outdegree strong components is used to characterize and detect deadlocks. Once a deadlock state is detected, an automatic procedure is initialized by using a reserved central buffer with unit capacity to resolve the deadlock. Other deadlock detection and recovery approaches based on graph theory can be found in [119], [121], [134], [231], [232], [241].

There is not much work on deadlock detection and recovery based on a Petri net formalism. Basile et al. [13] consider the supervisory control problem for a Petri net whose structure is known and its initial marking is fully or partially unknown. Markings are estimated by observing the occurrences of events that are assumed to be controllable. The inaccurate marking estimate can in theory lead to deadlocks that are called observer-induced deadlocks, which is decided by checking whether a particular deadlock condition is satisfied. When the controlled system satisfies the condition and consequently no further event is observed for a sufficiently long time, then an automatic deadlock recovery procedure is awakened. Similar ideas can be found in [61] and the related results on marking estimation and deadlock recovery are summarized in [79].

4 Deadlock Avoidance

The work in [213] is usually considered to be earlier and seminal research that demonstrates the use of Petri nets in modeling and deadlock control of FMS. More importantly, deadlock prevention and avoidance approaches are proposed. The former is achieved by an exhaustive path analysis of the Petri net model of the given FMS. It is feasible for a reasonably small system whose state space is small enough to fit in the memory of our computer. Deadlock avoidance is effectively implemented by an online monitoring and control system in which the Petri net model is used to look ahead into the future system evolution in order to make a correct deadlock-free resource allocation strategy. It is shown that deadlock avoidance is feasible for large real-world systems. However, deadlock states cannot be completely eliminated in general due to the limited number of look-ahead steps.

Motivated by the banker’s algorithm, a resource allocation and deadlock avoidance algorithm developed by Dijkstra [51] for operating systems, the work in [59] proposes a deadlock avoidance algorithm for a class of manufacturing-oriented Petri nets, called S*PR, denoted by $N = (P_0 \cup P_A \cup P_R, T, F)$, where $P_0$, $P_A$, $P_R$, and $T = \bigcup_{i=1}^{n} T_i$ are the sets of idle places, activity (operation) places, resources, and transitions, respectively. $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation between $P = P_0 \cup P_A \cup P_R$ and $T$. The complexity of the deadlock avoidance algorithm is $O(|P_A|^2 T_R |T_i^{\max}|)$, where $T_i^{\max} = \max\{|T_i| i \in \{1, 2, \ldots, n\}\}$. The algorithm is then used to a more general class of Petri nets, called non-sequential resource allocation processes [60].

Wu and Zhou have been active in the area of deadlock avoidance by using resource-oriented Petri nets (ROPN) as a formalism [230]. The main contributions in [219] consist of a new class of Petri nets, called colored resource-oriented Petri nets (CROPN), and necessary and sufficient conditions for deadlock-free operations from which a maximally permissive control law is derived. The deadlock-freedom of interactive subnets in a CROPN is verified by the control law. The whole CROPN of the system is shown to be live if each of its interactive subnets is controlled.
The work in [220] addresses the issue whether the maximally permissive policy proposed in [219] is optimal in terms of productivity in an environment where dispatching rules dominate and optimal scheduling is impossible due to either unaffordable computation required or changing operational parameters and structures. A novel deadlock control policy is developed such that it can avoid deadlock completely, and reduce starvation and blocking situations significantly by releasing an appropriate number of jobs into the system and controlling the order of resource usage based on state information in the net model. By using CROPN, the work in [222] handles the modeling, deadlock avoidance, and conflict resolution in AGV systems with bidirectional and unidirectional paths. The contributions in [223] are two-fold. One is the system modeling by integrating the models of the AGV system and part processing processes through macro transitions, where CROPN are used as a modeling paradigm. The other is the development of a deadlock avoidance policy. The policy is shown to be maximally permissive with computational complexity of $O(n^2)$ where $n$ is the number of machines in the system. The work in [224] aims to find shortest time routes for bidirectional AGV while both deadlock and blocking are avoided. Compared with other methods, the proposed one can offer better solutions. Also, its performance analysis is conducted via experimental studies.

Different from the traditional modeling approach for robots in an automated FMS, the study in [225] presents a novel point of view in modeling robots. Under such a modeling paradigm, an interesting and appealing result is obtained, i.e., the robots have no contribution to deadlock. Furthermore, the robots can be used to resolve deadlock by serving as temporary part storage devices. A new deadlock control policy is proposed by treating robots as both material handling devices and buffers, which is more permissive than the existing ones.

**Remark 1** The interesting and attractive proclamation made in [225] deserves to be carefully considered. It is known that a net model is a mathematical and formal description of a system. To fully understand a system, it is required that the model must sufficiently describe the physical characteristics of the system. Specifically, the problem that which resources can contribute to potential deadlocks in an FMS is completely decided by the configuration of its physical subsystem and its process planning of the parts to be produced, and is independent from the modeling styles and skills. It is reasonable that such a robot modeling treatment can facilitate the development of a deadlock avoidance policy. However, solid evidence that the policy is definitely more permissive that the existing ones in [58] and [135] remains to be found.

In our own opinion, such a modeling paradigm condenses the states in the traditional process-oriented models [228] used in most deadlock handling work such as [58] and [135]. Moreover, the modeling treatment of robots in [225] excludes some behavior of the plant, which is inadmissible from the deadlock control point of view. For example, a robot is not allowed to pick up a raw part if its serving machine tool has no capacity to host the part. Actually, some control ideas are mingled with the modeling in [225].

Behavior permissiveness is one of the most important criteria to evaluate a deadlock-free supervisor for an FMS. For a class of Petri nets, i.e., Systems of Simple Sequential Processes with Resources ($S^3$PR), Xing et al. [235] propose an optimal deadlock avoidance policy with polynomial complexity when an $S^3$PR satisfies a special condition with respect to its net structure and initial marking. Resource-transition circuits (RT-circuits), perfect RT-circuits (PRT-circuits), and maximal PRT-circuits (MPRT-circuits) are defined. A resource is said to be a $\xi$-resource if it is shared by two MPRT-circuits that do not contain each other. It is claimed that in an $S^3$PR without $\xi$-resources, the reachable markings are either safe or deadlocks. As a result, a one-step look ahead method can be utilized to avoid
deadlocks completely. It is shown that the supervisor for an $S^3PR$ without $\xi$-resources is maximally permissive, i.e., all deadlock states are eliminated and all safe states are kept in the controlled net system. In case of the existence of $\xi$-resources, a suboptimal supervisor can be obtained.

The work in [235] is extended by Wu and Zhou in [229]. They consider sequential resource allocation problems in automated manufacturing systems in which each processing step needs a single type of resources. The research proposes necessary and sufficient conditions under which a one-step look-ahead maximally permissive deadlock avoidance policy exists. Also, an approach is developed to derive such a policy if it exists. Note that in [229] a system is modeled with ROPN and in [235] a system is modeled with traditional process-oriented Petri nets. The former is claimed to be more compact and effective for deadlock resolution [228] than the latter. Similar results can also be found in [132], [187], and [66] in which automata or digraph are used to be a modeling paradigm.

The work in [3] contributes a hybrid approach that combines deadlock prevention and avoidance for a class of Petri nets, $S^4R$, which is more general than $S^3PR$ and allows multiple resource requirements in performing an operation on a part. When deadlock prevention phase cannot eliminate deadlocks, an online dynamic resource allocation policy is applied to the controlled system in order to minimize the probability of deadlock occurrences by trying to make always marked the monitors that results from the deadlock prevention. Even if both deadlock prevention and avoidance are employed, the occurrences of deadlocks remain possible in theory. Hence this method is not feasible in highly automated FMS. The idea in [3] can be improved by considering a polynomial deadlock prevention policy developed by Park and Reveliotis [171].

The deadlock avoidance algorithms reported by Roszkowska et al. in [6], [7], [189], [190] are usually considered to be a classical and seminal work in deadlock control by using Petri nets. Roszkowska’s work can be used for a class of Petri nets called PPN (Production Petri Nets) that is less general than $S^3PR$ proposed by Ezpeleta and his colleagues in 1995 [58]. The policy uses a feedback mechanism by monitoring the system state concerning currently active jobs to make a correct resource allocation decision by disabling some enabled transitions. It is shown that the policy is not optimal and in some cases is overly conservative.

Hsieh and Chang [89] propose a bottom-up approach to synthesize a controlled production Petri Net (CPPN) model of FMS, which consists of resource subnets, job subnets, and exogenous controls to describe the deadlock avoidance problem. Resource and job subnets are merged into a PPN that models the interactions among operations, resources and jobs in the FMS. Every transition is associated with a control place that controls its firing even if it is enabled in the sense of traditional net theory. Based on decomposing the CPPN into controlled production subnets of individual job types, a necessary and sufficient condition is developed for a CPPN to be live, which is motivated by the concept of minimal resource requirement at the idle state places to complete all the jobs in the CPPN. A sufficient validity test procedure with polynomial complexity is proposed to check whether the execution of a control action is valid to maintain the liveness of the CPPN and a synthesis algorithm is developed to determine a sequence of valid control actions to carry out the given dispatching policy. It is cogent that the proposed deadlock avoidance controller can achieve a high resource utilization.

Machine and device failure is inherent and common in many manufacturing systems. Currently, in the majority of automated manufacturing systems, the failure of a single device such as a workstation or even an individual sensor can cause the whole system to shutdown. Based on controlled assembly Petri nets (CAPN), Hsieh [90] develops a suboptimal deadlock avoidance algorithm with polynomial complexity for manufacturing
assembly processes with unreliable resources. Resource failure is modeled as loss of tokens in a Petri net model to represent unavailability of resources in the course of recovery procedures. Three types of token loss are defined to model 1) resource failures in a single operation, 2) resource failures in multiple operations of a production process and 3) resource failures in multiple operations of multiple production processes. Sufficient conditions are established for each type of token loss, which guarantee the liveness of a CAPN after some tokens are removed. An algorithm is proposed to conduct feasibility analysis by searching for recovery control sequences and to keep as many types of production processes as possible in production.

The significant contributions to the area of deadlock avoidance for automated manufacturing systems by using Petri nets as a formalism include Wu and Zhou [219]–[229], while Lawley, Reveliotis, Ferreira, and Fanti dominate the area of automata and graph theory to deal with deadlock control [52], [62]–[72], [125]–[133], [183], [188].

It is shown that optimal deadlock avoidance is NP-complete even for LIN-SU-RAS (Liner Single-unit Resource Allocation Systems) that is equivalent to PPN [187]. However, for DIS-SU-RAS (Disjunctive Single-unit Resource Allocation Systems), if the capacity of each resource is bigger than one, its optimal supervisory control can be done in polynomial time. A DIS-SU-RAS is actually equivalent to $S^3PR$.

5 Deadlock Prevention

From quantitative literature analysis, we find that deadlock prevention for FMS is investigated extensively and marvelously, leading to a vast stock of results. On one hand, the previous work on deadlock control has built a solid foundation for deadlock prevention research. On the other hand, a highly automated system cannot tolerate the occurrence of deadlocks that may result in severe economic loss or sometimes catastrophic results. This section reviews typical deadlock prevention strategies that are, by using Petri nets as a formalism, developed on the basis of different techniques, i.e., initial marking configuration, partial or complete reachability graph analysis, structural analysis, and combination of several methods.

5.1 Initial Marking Configuration

The research in the early stages that Petri nets are used to be a formalism to design deadlock prevention supervisors for FMS is close to the synthesis of the Petri net model of a system such that the model has desired behavioral properties such as liveness, boundedness, and reversibility. Petri net based modeling theory and approaches were the major topics at that time. The distinguished and representative researchers during the rough decade from 1985 to 1995 are Desrochers, DiCesare, Jeng, and Zhou.

The seminal work in [244] formulates a Petri net synthesis method of modeling automated manufacturing systems with shared resources. The concepts of parallel and sequential mutual exclusions are proposed. First, the places in a Petri net model of an automated manufacturing system are categorized into three classes: A-places, B-places, and C-places. An A-place is also called an operation place, representing a processing step of a raw part. It is unmarked at an initial marking, implying that no processing steps are activated in the initial state of a manufacturing system. B-places are called fixed resource places that are used to model manufacturing activity executors such as machine tools, robots, buffers, and part conveyors. A C-place is called a variable resource place that represents the availability of raw parts, fixtures, pallets, etc. At an initial marking, the number of tokens in a B-place or C-place is greater than zero. Based on the place partition, Zhou and DiCesare
find a relationship between the markings of B-places and C-places, under which the Petri
net model is live, bounded, and reversible.

In a decade following the work in [244], an enormous amount of work has been done
by DiCesare, Jeng, and Zhou [107], [108], [109], [111], [114], [115], [245], [246], [247]. Live-
ness conditions are established for a variety of manufacturing-oriented Petri net subclasses
such as PNR (Process Nets with Resources) [113], RCN(Resource Control Nets)-merged
Nets [110], ERCN(Extended ECN)-merged nets [233], which are usually expressed as the
relationships between the initial markings of B-places and C-places. The systematic mod-
eling techniques of Petri nets for automated manufacturing systems are gradually shaped.
Up to now, the Petri net modeling theory has basically become mature although some
work were reported in recent years [17], [226]. In summary, the research work in the early
stages of this direction combines modeling and control such that the resulting net model
has some desired behavioral properties such as liveness, boundedness, and reversibility.
The Petri nets considered in Zhou and Jeng’s work are usually ordinary, i.e., the weight
of an arc is one.

Figure 2(a) shows a small manufacturing system whose Petri net model is depicted in
Figure 2(b), where $p_1$ is a C-place, $p_5$ and $p_6$ are B-places, and the others are A-places. It is
verified that the net model in Figure 2(b) is live if $M_0(p_5) + M_0(p_6) > M_0(p_1)$. The initial
marking configuration for liveness is in some cases rather conservative and deteriorates the
performance of the controlled system. Take the net shown in Figure 3(a) as an example.
In order to keep liveness, $M_0(p_1) \leq 2$ is necessary if $M_0(p_5) = 2$ and $M_0(p_6) = 1$. In
this case, the resource represented by $p_8$ cannot be fully utilized. This approach tackles
the liveness problem by a way of global restriction, not focusing on a part or component
that results in deadlocks. Furthermore, deadlocks cannot be eliminated in some systems
by configuring an initial marking. For the net shown in Figure 3(b), one cannot find an
initial marking at which the net is live. However, its liveness can be enforced by properly
designing a monitor, as depicted in Figure 3(c).

![Figure 2: (a) A manufacturing system, and (b) its net model $(N, M_0)$]
Recent contributions in [11], [12], and [254] belong to this class of research mentioned above. The work in [12] deals with the verification of soundness, i.e., correct termination, of a procedure in workflow nets [1]. The soundness of a workflow net is tied to its liveness and boundedness. Based on workflow nets, a novel class of Petri nets, called workflow nets with resources, is proposed. A workflow net with resources can model rather general manufacturing systems with assembly and disassembly operations. Variations of a workflow net with resources include G-tasks [11] and G-systems [254]. Among the contributions by Barkaoui and his colleagues, one of the most important results is the controllability condition of a siphon, which is sufficient for the absence of dead transitions related to the siphon.

Let $S \subseteq P$ be a subset of places in a Petri net $N = (P, T, F, W)$ and $x \in P \cup T$ be a node. $\text{\*}x = \{y | (y, x) \in F\}$ and $x \text{\*} = \{y | (x, y) \in F\}$ are called the preset and postset of $x$, respectively. $\text{\*}S = \cup_{x \in S} \text{\*}x$ and $S\text{\*} = \cup_{x \in S} x \text{\*}$ are called the preset and postset of $S$, respectively. $S$ is called a siphon (trap) if $S \subseteq S\text{\*}$ ($S \text{\*} \subseteq S$). A siphon is said to be minimal if it contains no other siphons as its proper subset. A minimal siphon is strict if it contains no trap. The controllability of a siphon in a G-system is demonstrated by the following example.

Let $N = (\{i, o\} \cup P_A \cup P_R, T, F, W)$ be a G-system, where $i$ is called a source place with $\text{\*}i = \emptyset$, $o$ is called a sink place with $o \text{\*} = \emptyset$, $P_A$ is a set of activity places, and $P_R$ is a set of resource places. Let $P = \{i, o\} \cup P_A \cup P_R$. $F$ is the flow relation between nodes in $P \cup T$, and $W$ assigns a non-negative integer weight to any pair $(x, y)$ with $x, y \in P \cup T$. $\forall r \in P_R$, there exists a minimal P-semiflow $f_r : P \rightarrow \mathbb{N}$ with $f_r(r) = 1$ and $\forall p \in P_A$, there exists a minimal P-semiflow $f_p : P \rightarrow \mathbb{N}$, where $\mathbb{N}$ is a set of non-negative integers. Let $S$ be a minimal siphon and

$$gs = \sum_{r \in S \cap P_R} f_r$$

$$Out(S) = ||gs|| \setminus S$$
\[ h_S = \sum_{p \in \text{Out}(S)} f_p \]

\[ d_S = \max_{p \in \text{Out}(S) \cap \{||g_S||\}} g_S(p) \]

\[ z_S = g_S - d_S h_S \]

Siphon \( S \) is max-controlled if

\[ z_S^T M_0 > \sum_{p \in S} z_S(p)(\max_{p \bullet} - 1) \]

where \( ||g_S|| = \{p | p \in P, g_S(p) \neq 0\} \) is called the support of a \(|P|\)-vector \( g_S \) and \( \max_{p \bullet} = \max\{W(p,t) | t \in p \bullet\} \) is the maximal weight of the arcs from \( p \) to the transitions in its postset.

Figure 4: (a) A G-system \((N, M_0)\) and (b) its controlled net system \((N_1, M_1)\)

Take the net shown in Figure 4(a) as an example, where we assume that \( M_0 = j.i + k.r \), implying that source place \( i \) has \( j \) tokens, resource place \( r \) has \( k \) tokens, and the others are unmarked at the initial marking. Note that \( M_0 = j.i + k.r \) is a multi-set representation of vector \( M_0 \). According to the definition of G-systems, we should have \( k \geq \max\{f_r(p) | f_r(p) \neq 0\} = 5 \). It is verified that \( S = \{r, p_2, p_4\} \) is a strict minimal siphon. We hence have

\[ g_S = r + p_1 + 3p_2 + 2p_3 + 5p_4, \text{Out}(S) = ||g_S|| \setminus S = \{p_1, p_3\}, \]

\[ h_S = \sum_{p \in \text{Out}(S)} f_p = i + p_1 + p_2 + p_3 + p_4 + o, \]
$$d_S = \max_{p \in \text{Out}(S) \cap h_S} g_S(p) = \max\{g_S(p_1), g_S(p_3)\} = 2,$$
and
$$z_S = g_S - d_S h_s = (r + p_1 + 3p_2 + 2p_3 + 5p_4) - 2(i + p_1 + p_2 + p_3 + p_4 + o) = r + p_2 + 3p_4 - p_1 - 2i - 2o.$$

Considering that $M_0 = j.i + k.r$, i.e., $\forall p \notin \{i, r\}$, $M_0(p) = 0$, $\max r_{\bullet} = 3$, $\max p_{\bullet}^{p_2} = 1$, and $\max p_{\bullet}^{p_3} = 1$, $S$ is max-controlled if $k - 2j > 2$. From this example, the controllability of a siphon can be represented by a relationship between the initial marking of source and resource places. Clearly, $k = 5$ leads to $j = 1$ for $S$ to be max-controlled in Figure 4(a). The unique strict minimal siphon satisfies max-controlled siphon property, cs-property for short, implying the liveness of the G-system. A G-system satisfies the cs-property if there exists an initial marking such that the controllability of each siphon is ensured.

**Remark 2** The deadlock prevention based on the initial marking relationship of source and resource places is over-conservative in general. A monitor-based supervisor is shown in Figure 4(b) that is more permissive than the one in Figure 4(a) where the cs-property is ensured by the initial marking relationships.

### 5.2 Structural Analysis Methods

The major weakness of the deadlock prevention in which liveness is conditioned by an initial marking relationship between B-places and C-places is its conservativeness. As is known, a direct and significant consequence is that the productivity of a system can be deteriorated. In 1995, a seminal work was conducted by Ezpeleta et al. [58] who developed a design method of monitor-based liveness-enforcing Petri net supervisors for FMS, which is usually considered to be a classical contribution that utilizes structural analysis techniques of Petri nets to prevent deadlocks in FMS. First, a class of Petri nets, called $\text{S}^3\text{PR}$ (System of Simple Sequential Processes with Resources), is proposed and the relationship between strict minimal siphons and liveness of an $\text{S}^3\text{PR}$ is established. It is shown that an $\text{S}^3\text{PR}$ is live iff no siphon can be emptied. For each strict minimal siphon, a monitor is added such that it is controlled, i.e., cannot be unmarked at any reachable marking. After all siphons are controlled, the resulting net called controlled net system is live.

The significance of this approach is that it successfully separates a plant net model and its supervisor such that there is a clear boundary between them, as done in R-W theory of supervisory control for discrete event systems [179], [218] where automata are used to be a modeling formalism. The liveness-enforcing supervisor in [58] is a Petri net consisting of the monitors and the transitions of the plant model. Note that an $\text{S}^3\text{PR}$ can model automated manufacturing systems with flexible process routings but cannot model systems with assembly and disassembly operations since it is composed of state machines and resources. State machines are a typical Petri net subclass in which a transition has only one input and output place.

In general, the approach in [58] suffers from the following problems: behavioral permissiveness, computational complexity, and structural complexity. Behavioral permissiveness problem is referred to as the fact that permissive behavior is overly restricted by the deadlock prevention policy, i.e., the supervisor excludes some safe (admissible) states. This is so since the output arcs of a monitor are led to the source transitions of the net model, which limits the number of workpieces being released into and processed by the system. A source transition is the output transition of an idle place, which models the entry of raw parts into the system. Computational complexity results from the complete siphon enumeration that is necessary to compute a supervisor [58]. As known, the number of siphons grows fast and in the worst case grows exponentially with respect to the size of a
net model. Structural complexity is referred to as the number of monitors in a liveness-enforcing supervisor, which is in theory exponential with respect to the size of a net model since every strict minimal siphon needs a monitor to prevent from being emptied. The structural complexity of a supervisor means extra cost in system verification, validation, and implementation. Since 1995, much research work has focused on solving the above problems.

For a class of Petri nets, called PPN (Production Petri Nets), Xing et al. [234] propose a deadlock prevention policy. A transition in PPN is enabled if it is both process-enabled and resource-enabled. Then, the concept of deadlock structures is proposed, which is a set of transitions that are process-enabled but not resource-enabled. For each deadlock structure, a monitor is added such that a liveness-enforcing controlled system can be computed. Also, the concept of critical resources is formulated. It is shown that a maximally permissive, i.e., optimal, liveness-enforcing controlled net system can be obtained if the capacity of each critical resource is greater than one. However, the supervisor in [234] also suffers from computational complexity and structural complexity problems. The number of deadlock structures is in theory exponential with respect to the size of a PPN. It can be shown that a deadlock structure can derive a strict minimal siphon.

Wang et al. [215] show that the supremum of the strict minimal siphons in a PPN is $2^n - n - 1$, where $n$ is the number of resource places. As a result, the deadlock prevention policy needs in the worst case to add $2^n - n - 1$ monitors, leading to a supervisor with exponential complexity in structure with respect to the plant size. Its computational complexity problem results from the complete enumeration of deadlock structures. Although a PPN is a subclass of an $S^3PR$, the idea underlying the maximally permissive deadlock prevention policy is significant. Recently, it is extended to design maximally permissive supervisor for $S^3PR$ [147], [235].

Fairly speaking, both studies in [58] and [234] are of equal significance and seminality. From literature analysis, however, the work in [58] has more profound and far-reaching influence on the development of deadlock prevention methods based on Petri nets as a formalism. The reasons for this are twofold. On one hand, siphons, a well-defined and well-known structural object, are used to characterize and analyze deadlocks in [58] such that the policy is more understandable. Xing et al. [234] use a newly defined concept of deadlock structures. On the other hand, a more general class of Petri nets is defined.

Due to the inherent complexity of Petri nets, any deadlock prevention policy that depends on the complete siphon enumeration is definitely exponential with respect to the size of its plant net model. Given a Petri net, a maximal unmarked siphon can be obtained by the following traditional siphon solution. First, remove all the unmarked places. Then remove the transitions without input places as well as their output places. Repeat the two steps until no places and transitions can be removed. Chu and Xie [39] propose a deadlock detection method based on solving a mixed integer programming (MIP) problem, which is a mathematical programming implementation of the above traditional siphon solution approach. A feasible solution corresponds to a maximal unmarked siphon when there exists a siphon that can be emptied at a marking that is reachable from the initial marking. Otherwise, its optimal solution is equal to the number of all the places in the Petri net. Although an MIP problem is NP-hard in theory [74], [217], extensive numerical studies show that its computational efficiency is relatively insensitive to the initial marking and is more efficient than those that depend on complete state or siphon enumeration. Deadlock control is usually concerned with minimal siphons. Huang et al. investigate minimal siphon extraction methods from a maximal unmarked siphon [97], [140]. A software package that can do so is developed by Liu et al. [152]. A minimal siphon in a structurally bounded ordinary net can be directly found by solving an MIP
problem [80], [82]. Similar work on minimal siphon extraction using MIP is reported by Chao [29].

In order to tackle the computational complexity problem in [58], the MIP-based deadlock detection approach finds its further applications. Huang et al. develop an iterative deadlock prevention policy for S₃PR that consists of two phases. The first is called siphon control and the second is control-induced siphon control. At each iteration, a maximal unmarked siphon is computed by using the MIP-based deadlock detection method. Then, a minimal siphon is derived from the maximal one. A monitor is added such that the minimal siphon is controlled by the enforcement that the set consisting of the monitor and the complementary of the derived minimal siphon is the support of a P-semiflow. Repeat the above steps until all siphons in the original plant model are controlled. The second phase becomes necessary if the resulting net after the first phase contains deadlocks. At the second phase, a minimal siphon that contains at least a monitor is derived by solving an MIP problem. Then, a monitor is added to make the siphon controlled by leading the output arcs of the monitor to the source transitions of the plant model. This step is repeated until the MIP problem shows that there is no unmarked siphon at a reachable marking, implying that liveness is enforced. Experimental study shows that the two-phase policy is more permissive than the one in [58] and [135] although no formal proof is provided. However, there is some uncertainty in the number of reachable states of the controlled system. This is not surprising since in the second phase selecting different siphons to control can lead to liveness-enforcing controlled systems with different permissive behavior.

The two-phase deadlock prevention policy improves the work in [58] in a number of aspects. First, a more permissive supervisor can be obtained. Second, computational cost of a liveness-enforcing supervisor is reduced through the use of the MIP-based deadlock detection method. However, there is no definite improvement on the structural complexity of supervisors. Later, the ideas is applied to ES₃PR [98], a more general class of Petri nets than S₃PR.

The work in [27] proposes the concept of basic and compound siphons. First monitors are added to each basic siphon. Then it finds conditions for a compound siphon to be implicitly controlled. This research relieves the problem of siphon enumeration, which grows exponentially, and reduces the number of subsequent time-consuming MIP iterations.

Aiming at improving the behavioral permissiveness of a liveness-enforcing Petri net supervisor for an S₃PR, the work in [9] proposes the concept of basic and independent strict minimal siphons, which in fact motivates the development of elementary siphons in [135], [146]. For each basic or independent strict minimal siphon, a monitor is designed to prevent it from being unmarked. In a general case, the addition of monitors leads to control-induced siphons whose controllability is ensured by properly configuring the initial marking in the monitors for independent siphons. A case study shows that the proposed deadlock prevention policy is nearly optimal and more permissive than that in [58].

In order to model general systems, a variety of manufacturing-oriented Petri net subclasses are proposed such as AMG (Augmented Marked Graphs) [39], S⁴R (System of Sequential Systems with Shared Resources) [3], S⁴PR [200], WS³PR (Weighted System of Simple Sequential Processes with Resources) [199], PNR (Process Nets with Resources) [113], RCN (Resource Control Nets)-merged nets [110], ERCN (Extended Resource Control Nets)-merged nets [233], ERCN*-merged nets [114], S²LSPR (Systems of Simple Linear Sequential Processes with Resources) [170], S⁴PGR² (System of Simple Sequential Processes with General Resource Requirements) [171], S⁴PMR (System of Simple Sequential Processes with Multiple Resources) [99], G-tasks [11], WNSR(Workflow Nets with Shared Resources) [12], and G-systems [254]. From their definitions, S⁴R, S⁴PR, and S⁴PGR² are equivalent, which are named by different researchers. Similarly, ES³PR [98]
and S³PMR [99] are equivalent.

Most of the extended versions from PPN [7] and S³PR are generalized Petri nets in which there at least exists an arc whose weight is greater than one. Note that deadlock control in a generalized Petri net is much more difficult than that in an ordinary one. In an ordinary net, the weight of an arc is one. This implies that the transitions in the postset of a marked siphon will not be disabled totally. That is to say, there necessarily exist enabled transitions in the postset of the marked siphon. Due to this, an elegant result in an ordinary net is developed, which is invariant-controlled siphons [123]. A siphon is said to be invariant-controlled if it is a subset of the positive support of a P-invariant and the weighted token sum in the support of the invariant at initial marking is greater than zero. An invariant-controlled siphon can never be unmarked at any reachable marking from the initial marking [39], [115], [123], [124]. However, the weight of an arc in a generalized Petri net can be an arbitrarily given positive integer such that it is difficult to properly decide the lower bound of the number of tokens in a siphon.

Barkaoui and Pradat-Peyre are the pioneers who investigate the explicit siphon control problem in generalized Petri nets [10]. A siphon is called max-marked at a marking if the number of tokens in its a place is not less than the maximal weight of the arcs from the place. It is said to be max-controlled if it is max-marked at any reachable marking. A max-controlled siphon can at least fire once a transition in its postset. A Petri net is deadlock-free if all siphons are max-controlled, i.e., satisfying max cs-property (controlled-siphon property). Similar to the case of ordinary nets, a siphon’s controllability condition with respect to a P-invariant is developed. However, such a condition is sufficient but not necessary, and over-conservative when it is used to handle deadlock problems in some subclasses of manufacturing-oriented nets [20].

For siphons in S⁴R, Chao [20] relaxes the max-controllability condition by introducing max'-controlled siphons, which is motivated by the concept of deadly marked siphons [171]. Zhong and Li [243] refine the concept of max'-controlled siphons and propose a formal description for them. They point out that the marking of the considered resource places satisfying $M(p) \geq \max_{t \in S \cap \bar{S}} \{W(p, t)\}$ can guarantee that the siphon is sufficiently marked, where $\bar{S}$ is the complementary set of a siphon $S$. However, this controllability condition of siphons is still restrictive and conservative. Liu et al. [157] further relax it by introducing a new condition called max''-controllability condition for S⁴R. A siphon that is max-controlled means that it is max'-controlled and a max'-controlled siphon is accordingly max''-controlled. However, the converse is not true. The study in [157] concludes that an S⁴R net is live if each siphon is max''-controlled.

In the net shown in Figure 5(a), its unique strict minimal siphon is $S = \{p_2, p_4, p_5, p_6\}$. It is max-controlled since at any reachable marking, there exists a place $p$ in $S$ such that the number of tokens in $p$ is not less than the maximal weight of the arcs from $p$. In Figure 5(b), $S = \{p_2, p_4, p_5, p_6\}$ is a minimal siphon. After a feasible transition sequence $\sigma = t_1t_4$ fires, a marking $p_1 + p_3 + p_5 + p_7 + p_8$ is reached, at which it is not max-marked. Hence, it is not max-controlled. However, it is max'-controlled and the net is live. In Figure 5(c), $\{p_2, p_4, p_6, p_7, p_8\}$ is a minimal siphon that does not satisfy the definition of a max'-controlled one. However, it is max''-controlled and the net is hence live.

According to siphon controllability results in a generalized Petri net, Abdallah and ElMaraghy propose a hybrid deadlock control policy combing prevention and avoidance techniques for S⁴R [3]. The proposed deadlock control policy for S⁴R is feasible although there exist some minor technical issues that are reported by Chao [24].

Siphons are well recognized to be tied with deadlocks, which is true either in ordinary or generalized Petri nets. The fact is adequately represented by Reveliotis [185], [186], [187]. Actually, the siphon control problem in a generalized Petri net is not well addressed yet
Figure 5: (a) $S = \{p_2, p_4, p_5, p_6\}$ is max-controlled due to place $p_9$, (b) $S = \{p_2, p_4, p_5, p_6\}$ is max$'$-controlled, and (c) $S = \{p_2, p_4, p_6, p_7, p_8\}$ is max$''$-controlled.
as in ordinary nets and further efforts are necessary. Iterative deadlock control is a classic strategy in deadlock prevention. The original work is done by Lautenbach and Ridder \[124\], which targets bounded and consistent marked Petri nets. If there exists an arc from a place to a transition and its weight is greater than one, the corresponding transition is split by inserting new places and transitions such that the resulting net is ordinary. Then all uncontrolled siphons are computed, and controlled by adding monitors. Repeat the above steps until all siphons are controlled. Finally, the split transitions are merged. Such an iterative deadlock control policy is claimed to be maximally permissive. Note that redundant monitors are common in a controlled system resulting from an iterative deadlock control approach.

Tricas utilizes an iteration approach to prevent deadlocks for FMS \[200\], \[201\], \[202\], \[203\]. At each iteration step, a siphon is computed and controlled by a monitor. Such a process is continued until all siphons are controlled. For an S^4R, this class of iterative deadlock prevention policies is usually believed to converge at some step although it is not an easy job to provide a formal yet satisfactory proof. However, the convergence under an iterative siphon control for deadlock prevention is not difficult to imagine if a plant net is bounded, the initial marking is contained in a maximal strongly connected component in its reachability graph, and there exists a subgraph of the maximal strongly connected component such that a Petri net representation can be found for the subgraph that is considered to be a finite automaton. The major disadvantage is that such an iterative approach, in a general case, hardly leads to an optimal supervisor due to the immature siphon control techniques for generalized Petri nets if deadlocks are eliminated by means of the concepts of max-controlled \[10\], max'-controlled \[20\], or max''-controlled siphons.

Motivated by the work proposed in \[124\], \[175\], and \[176\], an iterative siphon control approach for deadlock prevention in a manufacturing-oriented Petri net model by Zhang and Wu \[242\] can lead to a maximally permissive Petri net supervisor by designing monitors to prevent siphons from being unmarked. At each iteration step, a siphon to be controlled is carefully selected by considering the number of tokens in its complementary set. Case studies show that such consideration can usually generate a structurally compact supervisor, i.e., the number of monitors is small. Note that the approach in \[242\] avoids the complete or partial marking enumeration of a plant model.

In \[154\] and \[155\], a polynomial-time deadlock prevention policy is proposed for a class of FMS with buffers. The idea is to partition those shared buffers for storing different types of parts, thereby destroying the circular-wait condition. However, this policy is not optimal. The research in \[156\] defines the key-resource/operation-place pairs of S^3PR and proposes a deadlock prevention policy based on this concept. The idea is that we always leave a part requiring key resources for some fixed manufacturing processes, thereby avoiding circular waits. This policy can guarantee that the number of control places is not greater than the number of resource places. However, it is not optimal and requires that the capacity of each key resource place is bigger than one.

Other active researchers in iterative deadlock prevention are Iordache and Antsaklis \[101\], \[102\], \[104\]. The feature of their research is that no structural constraints are enforced to a Petri net, i.e., their work is applicable to any net. Furthermore, uncontrollable and unobservable transitions are considered. As usual, an iterative approach consists of siphon calculation and control. The major problem is computational complexity since complete or partial siphon enumeration is necessary, along with incidental structural complexity. Their work notices that there exist redundant monitors in the supervisors when an iterative process terminates. However, the monitor’s removal condition is overly strict.

Craig and Zuberek \[45\] propose an efficient siphon-based deadlock detection method. It introduces the concept of equivalent siphons and develops two types of net transformation
that can reduce equivalent siphons, making the siphon-based deadlock detection much more attractive from a practical point of view.

In addition to the MIP-based deadlock detection approach that is shown to be, although it is NP-hard in theory, computationally competitive via numerical examples, another significant yet computationally efficient breakthrough in designing liveness-enforcing Petri net supervisors is a deadlock prevention algorithm with polynomial complexity. Proposed by Park and Reveliotis [169], [171], the thoughts of this algorithm were originally derived from the seminal work [183], [184] in an automaton formalism. In other words, the work in [171] can be considered as a Petri net implementation of the deadlock avoidance algorithm in [183]. The deadlock control specifications are represented by a set of constraints. In a constraint, the number of tokens in a set of activity places is limited to be a constant. In fact, the deadlock prevention requirements and specifications are converted into a set of GMEC (Generalized Mutual Exclusion Constraints) that can represent the concurrent use of a finite number of resources shared among different processes [76], [78]. Each GMEC is implemented by a monitor. The number of monitors in the supervisor is equal to that of resources in the system under consideration. As a result, another advantage of this approach is that the structural complexity problem of a supervisor is well addressed. However, it suffers the behavior permissiveness problem. Experimental studies show that it is even less permissive than the policy in [58]. Moreover, resource ordering needs to be assigned before the computation of a supervisor. Different resource orderings lead to supervisors with different permissive behavior. In this sense, selecting an optimal resource ordering is of importance to behavior permissiveness.

The study in [151] proposes the concept of dominated transitions. It indicates that the output arcs of a monitor are not necessarily led to the source transitions of a plant net model as done in [58], which is the reason behind the loss of many permissive states. Dominated transitions are useful to develop a more permissive liveness-enforcing supervisor than the policies in which monitors are led to the source transitions.

Giua and Seatzu discuss the liveness enforcement problem in railway networks by using Petri nets [82]. A supervisor is found by adding appropriate monitors designed through siphon analysis. It does not need an exhaustive computation of all siphons. Furthermore, uncontrollable and unobservable transitions in a plant model are considered. The work in [72] considers monitor design for colored Petri nets with applications to deadlock problems for railway networks.

Another interesting field is Internet-motivated video streaming systems, where deadlocks or blocking are caused by network resources with a high-sharing degree [92]. The contributions in [92] are twofold. First a novel class of Petri nets, called non-sequential systems of simple systems with shared resources, and their liveness analysis methods are proposed. Second, a deadlock prevention policy based on generalized elementary siphons is developed.

5.3 Reachability Graph-based Approaches

As stated previously, behavioral permissiveness is one of the most important criteria in evaluating a supervisor. Determining how to design a maximally permissive, i.e., optimal, Petri net supervisor has been an interesting yet significant problem, from both practical and theoretical points of view. On one hand, the problem is well addressed in the framework of formal languages and automata in R-W theory [218]. On the other hand, it has remained open in Petri net formalisms for many years. The reasons are twofold. First, Petri nets have limited modeling power, compared with automata. Specifically, it is not true that any automaton can find its free-labeled Petri net implementation. Second, Petri
Netters do not find effective approaches to deal with this problem except some special subclasses under a particular initial marking [234], [146].

The work by Uzam [205] proposes an approach to design optimal Petri net supervisors by using the theory of regions [5] that originally aims to provide a formal methodology to synthesize a Petri net from a transition system. It establishes a connection between transition systems and Petri nets through net synthesis. The idea behind the theory of regions is that a state-based model, a model describing which states a process can be in and which transitions are possible between these states, can be transformed into a Petri net, a compact representation of the state space, explicitly showing causality, concurrency and conflicts between transitions. Since its appearance, the theory of regions finds its wide and extensive applications [46]. Shortly after the work in [205], by using popular and plain linear algebra, Ghaffari et al. present an easily understandable explanation of the design approach to an optimal liveness-enforcing Petri net supervisor based on the theory of regions [75]. An improved version of the work in [205] can be found in [206].

Based on the theory of regions, a design approach of optimal liveness-enforcing Petri net supervisors can be expounded as follows. First, one generates the reachability graph of a plant Petri net model and then finds all marking/transition separation instances (MTSI) as well as the sets of legal and illegal markings. An MTSI takes the form of \((M, t)\), where \(M\) is marking, \(t\) is a transition, and \(M[t]\) is a forbidden marking that is illegal from the deadlock control point of view. For an MTSI \((M, t)\), a monitor is computed by solving a linear programming problem (LPP) such that its addition to the plant model disables \(t\) at \(M\) while ensures the reachability of all legal markings. The fatal disadvantage of the approaches based on the theory of regions is that complete state enumeration is necessary. As known, the size of the reachability graph of a Petri net grows quickly and in the worst case grows exponentially with respect to the number of its nodes and initial marking. This is the so-called state explosion problem.

Finding an MTSI can be done in polynomial or even linear time by a depth or breadth first search algorithm after the reachability graph is computed. However, for deadlock control purpose of a Petri net, the number of MTSI is in theory exponential with respect to the size of the model and its initial marking. Hence, the number of LPP to be solved is in theory exponential with respect to the plant net size. In this sense, polynomial solvability of an LPP seems meaningless. Moreover, in such an LPP, the number of constraints is almost equal to that of markings in the state space. Note that the number of LPP to be solved in theory equals to that of MTSI. However, a monitor can implement multiple MTSI, leading to the fact that the number of monitors is generally much smaller than that of MTSI.

It is worthy of noting that even before the publication of the work by Uzam and Ghaffari, a common sense is that any approach depending on complete state enumeration is infeasible to real-world production systems. The approaches in [75] and [205] can find an optimal supervisor if such a supervisor exists. When an optimal net supervisor does not exist, the work in [75] and [205] does not offer a deadlock control solution. For example, one cannot find an optimal Petri net supervisor for the net model shown in Figure 6 by using the theory of regions. In this case, a natural and interesting problem is to find a best permissive liveness-enforcing Petri net supervisor such that there are no other Petri net supervisors that are more permissive than it.

The work in [35] presents a deadlock prevention approach to find a maximally permissive liveness-enforcing Petri net supervisor for an FMS if such a supervisor exists. Otherwise, it can derive a best permissive liveness-enforcing Petri net supervisor in the sense that there do not exist other Petri net supervisors that are more permissive than it. The proposed approach only computes the reachability graph of a plant model once. By using the
vector covering approach, the sets of legal markings and first-met bad markings (FBM) are reduced to be two further small sets, namely, the minimal covering set of legal markings and the minimal covered set of FBM, respectively. At each iteration, an FBM from the minimal covered set is selected. By solving an integer linear programming problem (ILPP), a place invariant (PI) is designed to prevent the FBM from reachable and no marking in the minimal covering set of legal markings is forbidden. If the ILPP has no solution, implying that there is no maximally permissive Petri net supervisor for the plant net model, another ILPP is designed to remove the least number of legal markings whose reachability conditions contradict others. Then, a PI is redesigned to keep the rest legal markings being reached. This process is carried out until no FBM can be reached. Finally, a best permissive liveness-enforcing supervisor is obtained.

Figure 6: A Petri net without an optimal net supervisor.

Based on the theory of regions, Uzam and Zhou develop an iterative deadlock prevention approach [208]. It is assumed that there is a monitor solution to the deadlock prevention for a plant net model. The reachability graph is divided into two parts: deadlock-free zone and deadlock zone. A deadlock-free zone is in fact the maximal strongly connected component that contains the initial marking. A deadlock zone contains markings from which the initial marking is unreachable. An FBM is defined in the deadlock zone. It does not satisfy deadlock-free control requirements and its father node is in the deadlock-free zone. Then, deadlocks can be eliminated by preventing the firing of the transitions enabled at an FBM’s father marking such that the FBM is unreachable. It is shown that the transition firing control can be converted into a generalized mutual exclusion constraint (GMEC) problem that can be implemented by a monitor [237], [238] whose computation is highly efficient. Timed GMEC problems are considered in [96].

The approach in [208] suffers from two problems. On one hand, an optimal supervisor cannot be found in general even if such an optimal one exists. On the other hand, at each iteration step, one needs to compute the reachability graph once. It is claimed by the authors that the approach in [208] are efficient. However, its efficiency is naturally questioned since at each iteration step, the computation of a reachability graph is necessary to verify whether markings in deadlock zone are reachable. Note that the work in [205] and [75] needs to compute the reachability graph only once and then to solve LPP. The work in [207] aims to improve the deadlock prevention policy by reducing the plant net model as a first step before designing its optimal supervisor.

It is worthy of noting that a reachability graph is a reliable, accurate, and effective (surely not efficient) analysis method of a Petri net although it is computationally expensive. In recent years, abundant contributions have been seen on efficient computation and compact representation of a reachability graph, e.g., the state space representation based on binary decision diagrams (BBD) [15], [40], [160], [172], data decision diagrams [43], hierarchical set decision diagrams [44], [85], stubborn sets [210], [211] and sleep set
methods for reduced state space generation [212].

Chen and Li [34] present a novel and computationally efficient method to design optimal control places and an iterative approach that only computes the reachability graph once to obtain a maximally permissive liveness-enforcing supervisor of FMS. By using a vector covering approach, the minimal sets of legal markings and FBM are computed. At each iteration, an FBM from the minimal set is selected. By solving an ILPP, a PI is designed to prevent the FBM from being reached and no marking in the minimal set of legal markings is forbidden. This process is carried out until no FBM can be reached. In order to make the considered problem computationally tractable, BDD is used to compute the sets of legal markings and FBM, and solve the vector covering problem to obtain the minimal sets of FBM and legal markings. It is worthy of noting that this work offers an optimal supervisor for a large Petri net model with 48 places and 38 transitions by computing all reachable markings.

The work in [216] can be considered as an improvement of the theory of regions. It first designs a supervisor for a plant net model by using the theory of regions to find maximally permissive behavior. Then, the strict minimal siphons in the maximally permissive controlled system are computed and divided into elementary and dependent ones. To prevent them from being emptied, algebraic expressions about the markings of the additional monitors in the supervisor and the resource places in the plant net model are derived, under which the supervisor is live. The expressions are used to derive the live initial markings for the supervisor without changing its structure when the initial marking of the plant changes. A case study shows that the combined method is computationally efficient compared to existing ones in which the theory of regions is used alone, and the permissive behavior of the supervisor is near-optimal.

Piroddi et al. [175] believe that there are several important drawbacks in the deadlock prevention methods that are based on elementary siphons, a novel concept in net theory, that are originally developed by Li and Zhou [135], [143], [145], [146]. First, elementary siphons are developed by purely utilizing the topological structure of a net, not taking into account of the dynamical evolution information of the net. Second, the policies based on elementary siphons are usually not maximally permissive. Third, the set of elementary siphons in a Petri net is not unique. Deciding how to select a set of elementary siphons is important since the selection determines the sets of strongly and weakly dependent siphons. The existence of different sets of elementary siphons also implies that the deadlock prevention solution is not unique. Last but not least, the policies based on elementary siphons can be applied to some special classes of Petri nets only. Piroddi et al. think that it is necessary and reasonable to combine the structural information related to strict minimal siphons and reachability analysis that represents the dynamical evolution information in order to reduce the number of iterations in siphon control and the number of monitors.

The work in [175] develops a selective siphon control policy in which the concepts of essential, dominated, and dominating siphons and critical, dominating and dominated markings play an important role. By solving set covering problems, dominating siphons are found to ensure that dominated siphons are controlled. The idea to some extent is similar to that of elementary siphons although that their development is not necessarily motivated by the concept of elementary siphons. However, it considers the relationship between uncontrolled siphons and deadlock markings such that it becomes an accurate deadlock control method. On one hand, it needs less monitors than the existing methods do. On the other hand, the resulting supervisor is highly permissive. Although it is not formally proved to be optimal, it is at least near-optimal. It is worthy of noting that the work provides an optimal liveness-enforcing supervisor for a well-known FMS example originally presented in [58], which is computed within a reasonable time. Before this work,
the best but not optimal supervisor for this example is offered by Uzam and Zhou in [207] and [208].

The major technical problem in [175] is computational complexity. At each iteration, it needs to compute all minimal siphons and all dominating markings and to solve a set covering problem, each of which is NP-hard in theory with respect to the net size. In the computational overhead aspect, the comparison between the policy based on the theory of regions and this one remains open. After the complexity problem is recognized, Piroddi et al. shortly improve the method [176] by using the MIP-based deadlock detection approach such that the complete minimal siphon enumeration is avoided. From the case study, the improved version of the combined siphon and marking policy is computationally competitive.

Hu and Li [93] study the deadlock prevention problem for a class of conjunctive/disjunctive resource allocation systems (C/D-RAS) in which multiple resource acquisitions and flexible routings are allowed, representing a large class of automated manufacturing systems. Local and global control approaches can be derived. Compared with traditional methods reported in the literature, the proposed methods can well keep balance among computational complexity, behavioral permissiveness, and structural complexity. The local control approach can lead to a nearly maximally permissive liveness-enforcing Petri net supervisor with reasonable computational efficiency. The global control approach can result in a less permissive liveness-enforcing Petri net supervisor but with simple structural configuration. Moreover, both the local and global supervisors involve only limited online computation complexity. Similar work that avoids the complete siphon or state enumeration can be found in [94].

5.4 Elementary Siphon-based Approach

Even in a Petri net with a rather simple structure such as linear S$^3$PR (LS$^3$PR), the number of its strict minimal siphons is proved to be, in the worst case, exponential with respect to its size [215]. If all the strict minimal siphons are explicitly controlled without considering any difference among them, the resulting liveness-enforcing Petri net supervisor is structurally complex in theory, as shown in [58]. For example, an automated manufacturing system with 16 machine tools and 13 robots is considered. Its Petri net model consists of 128 places and 88 transitions. According to the deadlock prevention policy in [58], the supervisor contains 587 monitors and 15,464 arcs. One can imagine its huge structural complexity.

To alleviate such a structural complexity problem, Li and Zhou propose the concept of elementary siphons [135], [137], [146]. For a deadlock control purpose, problematic siphons (that can cause deadlock transitions) in a Petri net are divided into elementary and dependent ones. The latter is originally named as redundant siphons [135]. By the incidence matrix of a Petri net, Li and Zhou define the characteristic T-vector of a siphon, which is the sum of the rows corresponding to the places in the siphon, and indicates the change of the number of tokens in it when a transition fires. The characteristic T-vectors of all problematic siphons constitute a matrix called their characteristic T-vector matrix. From the matrix, a maximal linearly independent set of vectors can be found. The siphons corresponding to the vectors in this set are called elementary siphons. The others are called dependent siphons that are further distinguished into weakly and strongly dependent ones by deciding whether the linear combination coefficients are all positive or negative.

Two key contributions underlying the concept of elementary siphons are as follows: (1) The number of elementary siphons in a Petri net is bounded by the smaller of the place and transition counts and (2) A dependent siphon can be controlled by controlling
its elementary ones. For deadlock prevention achieved by monitors, it is of significance that a dependent siphon can be controlled via explicit control of its elementary siphons by designing monitors and properly setting their control depth variables. An FMS example is investigated in [135] with its Petri net model having 26 places, 20 transitions, and 18 strict minimal siphons. By using the concept of elementary siphons, a liveness-enforcing Petri net supervisor is computed by explicitly adding only six monitors to control the six elementary siphons among 18 strict minimal siphons. However, the work in [58] needs to design monitors for all 18 strict minimal siphon.

In [135], the controllability of a dependent siphon is explored with respect to elementary siphons that are invariant-controlled. A more general controllability condition than that in [135] is developed in [145] for dependent siphons in an ordinary Petri net. In [143] Li and Zhao extend such results to generalized nets, which is based on the max-controlled siphons [10]. However, computational complexity of supervisor design in [145] remains to be exponential with respect to the net size since the computation of a set of elementary siphons depends on the complete siphon enumeration.

The computational efforts for a supervisor based on elementary siphons are reduced by introducing a siphon solution technique using an MIP-based deadlock detection method. The study in [139] proposes an iterative deadlock prevention policy by using the concept of elementary siphons. At each iteration, a maximal unmarked siphon is found by solving an MIP problem. Then a strict minimal siphon is extracted from the maximal unmarked one. If the siphon extracted is elementary with respect to the computed ones, it is explicitly controlled by a monitor. If it is dependent, its controllability is decided by checking whether it needs to be explicitly controlled. The iteration process terminates when no unmarked siphon is found in the controlled Petri net with monitors.

The work in [139] to a large extent reduces the computational complexity to design a supervisor and the resulting supervisor’s structural complexity compared with [58]. However, it does not improve the behavior permissiveness. The work in [136] develops a two-phase deadlock prevention policy. The first phase adds a monitor for each elementary siphon derived from the MIP-based deadlock detection method. The output arcs of a monitor are led to the source transitions of a plant Petri net model, which represent the entry of raw parts into a system. The second phase rearranges the output arcs of the monitors such that they are as far as possible led to non-source transitions if this rearrangement does not result in dead transitions. Such an improvement increases the behavior permissiveness of a supervisor. The deadlock prevention policies underlying the idea of elementary siphons can also be found in [138], [240].

Hu and Li [95] extend the study in [139] to a more general class of Petri nets, S^4PR. Insufficiently marked siphons are used to characterize deadlocks, which can be found by solving an MIP problem. A generalized net is first transformed into an ordinary one before the MIP problem proceeds.

The work in [148] can be considered as an application of the divide-and-conquer strategy in deadlock control area for resource allocation systems, which is an important problem solution paradigm in computer science. A plant net model is divided into an idle subnet, an autonomous subnet, and a number of small but independent subnets, called toparchies, by the concept of resource circuits. A liveness-enforcing supervisor, called a toparch, is designed for each toparchy. If a particular separation condition holds in a plant net model, the computational overhead of toparches is significantly reduced. This research shows that the resulting net by composing the toparches derived for the toparchies can serve as a liveness-enforcing Petri net supervisor for the whole plant model. A case study shows the significance of the divide-and-conquer strategy via a number of typical deadlock prevention policies.
It is not surprising that there exist redundant monitors in a liveness-enforcing Petri net supervisor. In [209], a redundant monitor is identified and eliminated by computing the reachability graph of a controlled system. If the removal of a monitor does not lead to the loss of liveness of the controlled system, it can be removed from the supervisor. The major drawback is the computational complexity problem since the reachability graph and liveness check are necessary. In order to avoid the complete state enumeration, Li and Hu [149] propose two methods to remove monitors from a liveness-enforcing Petri net supervisor. The first method is based on the concept of implicit places [73], [181]. It is shown that the implicity of a monitor is decided by solving an LPP that can be done in polynomial time. The second one is derived from the MIP-based deadlock detection method. If the removal of a monitor does not change the optimal solution of an MIP problem derived from the controlled system, then it is implicit or its removal may lead to more permissive behavior while liveness is preserved.

As known, dependent siphons can be further divided into strongly and weakly dependent ones. An interesting work is done by Chao and Li in [26], which explores the structural condition in a class of Petri net under which there exists a set of elementary siphons such that all the others are strongly dependent.

The solution for elementary siphons in a Petri net is significant to the development of deadlock prevention policies. By using the concept of handles and bridges [56], much work is done by Chao on the computation of minimal and elementary siphons in a resource allocation system [18], [21], [22], [23], [25], [29], [142]. The work in [158] presents a polynomial algorithm to find a set of elementary siphons in S^4PR. The algorithm proposed by Wang et al. [215] is also of polynomial complexity, which can find a set of elementary siphons for linear S^3PR. A significant result in [215] is the supremum of the number of strict minimal siphons in a linear S^3PR, which is \(2^n - n - 1\) where \(n\) is the number of resource places.

A known result on deadlock-freedom in net theory is Commoner’s theorem, i.e., an ordinary net is deadlock-free if there is no siphons that can be unmarked. In many subclasses of Petri nets, the fact that there is no unmarked siphon implies their liveness. The subclasses include PPN [7], S^3PR [58], ES^3PR [98], and ERCN-merged nets [233]. The work in [19] proposes the concept of virtual first-order structures. A net without virtual first-order structures has the property that the absence of unmarked siphons implies the liveness. Furthermore, it is improved in a recent work [28] by showing that a non-virtual-net is live as long as all its siphons can never be emptied, which generalizes the existing net subclasses that has such a property. Other deadlock prevention policies developed by Chao can be seen in [30] and [31].

The concept of elementary siphons aims at simplifying the structure of a liveness-enforcing Petri net supervisor. The number of transitions in a supervisor is not greater than that of a plant model. As a result, structural complexity of a supervisor is usually evaluated by the number of its monitors. However, in a general case, a deadlock prevention policy based on elementary siphons cannot find a minimal supervisory structure. The work by Chen and Li in [36], by solving mixed integer programming problems, find a maximally permissive liveness-enforcing supervisor that has a minimal number of monitors under the condition that each monitor is associated with a P-semiflow whose support contains the monitor and some operation places. For the well-known FMS example whose Petri net model has 26 places and 20 transitions [58], an optimal supervisor with a minimal number of monitors, i.e., five, can be found.
5.5 Combined Techniques

To reduce the number of marking/transition separation instances when using the theory of regions, Li et al. develop a two-phase deadlock prevention policy by siphon control and the theory of regions [144]. First, strict minimal siphons are identified through resource circuits only and controlled by monitors, whose quantity is bounded by the smaller of place and transition counts. This lead to the fact that siphon identification and control is of polynomial complexity. Then, the theory of regions is applied to the augmented Petri net with monitors to find a supervisor. Since the siphon control in the first phase is optimal from deadlock prevention point of view, the final supervisor is optimal if such a supervisor exists.

Motivated by the tight connections between directed graphs (digraphs) and Petri nets in deadlock control for FMS [67], Maione and DiCesare [159] propose a hybrid deadlock prevention policy by using directed graphs and Petri nets, taking advantages of the strong points of both techniques. In order to avoid searching siphons in the Petri net model of a system, deadlock detection is carried out by analyzing the structures of the digraph that models the system. Then, the digraph-based information is translated into the deadlock marking of the corresponding Petri net, which is used to design monitors to prevent empty siphons. Finally, a number of new control policies are developed, which are less restrictive than other efficient policies [159].

The work in [158] presents a novel deadlock detection approach for WS3PR, an extension of System of Simple Sequential Processes with Resources (S3PR) with weighted arcs. It explores the numerical relationships among weights, and between weights and initial markings based on simple circuits of resource places, which are the simplest structure of circular wait, rather than siphons. It is shown that a WS3PR is deadlock-free and live if it satisfies a proposed condition with respect to arc weights and initial markings. A set of polynomial algorithms are developed to implement the proposed method.

6 Open Problems

6.1 Uncontrollable and Unobservable Transitions

Uncontrollable and unobservable events in a plant may be present. Accordingly, it is reasonable and practical to consider their existence in a Petri net model of an FMS. Note that in RW-theory [179], [218], uncontrollable and unobservable events are sufficiently considered. However, Petri net researchers usually assume that all transitions are controllable and observable when a deadlock prevention policy is developed for an FMS. When the presence of uncontrollable and unobservable transitions is taken into account, most existing deadlock control policies need to be refined or even reinvestigated. For example, an optimal supervisor cannot be found if some transitions are uncontrollable in a Petri net model [150].

The work in [101] considers the design of a liveness-enforcing supervisor for a Petri net with uncontrollable and unobservable transitions. A siphon’s controllability is represented by a linear constraint on markings. Two conditions are proposed to decide the admissibility of a linear constraint. An admissible constraint is enforced directly and an inadmissible one can be performed after it is transformed to be admissible by using the constraint transformation method in [162] and [163]. However, the deadlock prevention policy in [101] is performed in an iterative way. Also it suffers from a computational complexity problem since at each iteration step, complete siphon enumeration is needed.

The work in [178] considers the applicability of a deadlock prevention policy developed under the assumption that all transitions are controllable and observable, to a plant
with uncontrollable and unobservable transitions. The concept of critical controllable and observable transitions is proposed. It is concluded that a deadlock prevention policy developed without considering uncontrollable and unobservable transitions is applicable to a plant with uncontrollable and unobservable transitions if and only if those in the set of critical controllable transitions are controllable and those in the set of critical observable transitions are observable.

6.2 Fault-tolerant Deadlock Control

The selection of deadlock control strategies depends on the frequency of deadlock occurrences in a system. If deadlocks are rather rare, a time-out mechanism may be accepted as the best approach to deal with deadlocks due to its low overhead. This is deadlock detection and recovery. In some cases, this strategy is not permitted due to technical or other factors. Instead, deadlocks are expected to forbid even if some resources breakdown. In a contemporary manufacturing system, automated equipment is widely and extensively used. The occurrences of faults in unreliable devices and machines can falsify a correctly designed deadlock prevention policy [133]. Robust deadlock prevention and avoidance policies considering various errors and faults in an automated FMS are an interesting topic by using Petri nets as a formalism [90].

6.3 Existence of Optimal Supervisors

The existence of marking-based, not monitor-based, liveness-enforcing supervisors for discrete event systems is investigated by Sreenivas [196], [197], [198] in which the computation of the reachability graph is necessary. The problem is also discussed by Giua and DiCesare from the formal language point of view [77]. Iordache and Antsaklis develop generalized conditions for liveness enforcement and deadlock prevention in Petri nets [100], which are based on the concept of active subnets and siphons. However, no sufficient attention is paid to the existence of an optimal (monitor-based) liveness-enforcing Petri net supervisor for an FMS. A natural and interesting problem is the structural and initial marking conditions of a Petri net under which there exists an optimal liveness-enforcing Petri net supervisor. For example, is there an optimal supervisor for any $S^3PR$. For the existing manufacturing-oriented ordinary Petri net subclasses in the literature such as PPN [7] and $S^3PR$ [58], we have not seen any example whose optimal liveness-enforcing net supervisors do not exist. However, it is easy to find an $S^3PR$ whose optimal supervisors represented by Petri nets do not exist. The net shown in Figure 6 is such an example. However, there exists an optimal liveness-enforcing supervisor that takes the form of an automaton. In fact such an automaton cannot admit a free-labeled Petri net representation.

6.4 Deadlock Prevention under Dynamic Control Specifications

In supervisory control of discrete event systems, an important and typical class of control specifications is linear inequalities on markings of a Petri net [105], which is the so-called generalized mutual exclusion constraints (GMEC) [77]. Other forms of control constraints can be transformed into GEMC problems [103]. The control specifications are assumed to be stable during the design and runtime phases of a supervisor. However, in practice, control requirements can change during runtime phase of a supervisor. For example, they can vary due to breakdown of a machine tool or client order change. As far as the authors know, no attention is paid to the deadlock control problem in an FMS under dynamic control specifications.
Concluding Remarks

Absence of deadlocks is critical in systems that are expected to operate in an automated way. They include life-support systems, nuclear plants, transportation control systems, and automated manufacturing systems. A systematic and efficient method to prevent, avoid, and detect deadlocks is of primary importance for them.

With the wide application of agile and flexible production mode, ignoring deadlocks in an FMS is usually not an option. Over the past two decades, deadlock control research has received much more attention from academic and industrial communities, leading to ample resolution methodologies, most of which are based on a Petri net formalism. This research aims to present a literature review of deadlock control strategies for FMS with focus on deadlock prevention.

As a structural object, siphons are tied to dead transitions whose existence leads to the loss of liveness of a Petri net. Siphon control provides an effective way to prevent the occurrence of deadlocks. In a general case, a siphon-based deadlock prevention policy cannot find an optimal, i.e., maximally permissive, supervisor, particularly for generalized Petri nets. However, it can represent the liveness requirements as a set of linear inequality constraints with respect to initial markings. For a fixed net structure with an initial marking, once the liveness requirements are established, it is easy to decide its supervisor when the initial marking changes.

It is shown that reachability graph-based approaches can find an optimal supervisor if it exists. However, they usually suffer from expensive overhead since the complete state enumeration of a Petri net is exponential with its size and initial marking. The deadlock prevention policies that use partial reachability graphs seems to provide a trade-off between computational cost and behavioral permissiveness. For a fixed net structure with a new initial marking, all the computation needs to be carried out afresh since a reachability graph is sensitive to both net structure and initial marking. This is proved to be computationally inferior in comparison with siphon-based strategies since siphons are a pure structural objects whose computation is independent of initial markings.

Deadlock prevention through a proper configuration of initial markings is overly conservative and not recommendable in practice. Furthermore, it does not improve computational efficiency to design a supervisor. A hybrid deadlock control approach refers to the one that either combines deadlock prevention and avoidance or utilizes two different formal paradigms. The essential motivation of such a policy is to take the advantages of multiple strategies or formalisms, while avoiding their disadvantages. This paper should facilitate engineers in choosing a suitable deadlock control method for their industrial application cases.

Bibliography Notes

to discrete event systems can be found in [16] and a book on resource allocation system management from discrete event system point of view can be found in [187].

The survey papers in journals are [165] and [173] on basics of Petri nets and their analysis techniques, [192] and [195] on Petri nets and production systems, [53] on Petri nets in manufacturing system control, [141] on the comparison of deadlock prevention policies that are based on Petri nets via a typical case study, [71] on deadlock control methods in automated manufacturing systems, [164] on modeling FMS using Petri nets, and [86] on Petri nets as a formalism for discrete event systems.

Many papers in special issues in journals, or special sessions in international conferences or workshops are devoted to Petri nets and manufacturing such as [33], [48], [49], [112], [116], [117], [120], [193], [250], [252], [253], and [255]. Survey papers on Petri nets from a system theory perspective can be found in [81] and [194].

As a structural object, siphons play an important role in the analysis of structural and behavioral properties of Petri nets. An algorithm with polynomial complexity to decide whether a set of places is a minimal siphon can be found in [8]. Classic and typical siphon computation methods are presented in [57], [122], and [239]. A siphon computation method that is claimed to be rather efficient is developed in [42], which can find $2 \times 10^7$ siphons within one hour. For a class of Petri nets, a siphon solution approach is given in [2] that is also efficient through experimental studies. A parallel solution to compute siphons is established by Tricas and Ezpeleta [204]. An efficient minimal siphon computation approach by using BDD is provided in [37].

Acknowledgments
This work is supported by the National Natural Science Foundation of China under Grant No 60474018, 60773001, and 61074035, the National Research Foundation for the Doctoral Program of Higher Education, Ministry Education of China under Grant No 20070701013 and 20090203110009, “863” High-tech Research and Development Program of China under Grant No 2008AA04Z109, Chang Jiang Scholars Program, the Ministry of Education, P. R. China, Fundamental Research Funds for the Central Universities under Grant No. 72103326, the Ministry of Education, the Research Fellowship for International Young Scientists, National Natural Science Foundation of China, under Grant No. 61050110145, and Alexander von Humboldt Research Grant, Alexander von Humboldt Foundation, Germany. The authors would like to thank Yuh Chao, WeiMin Wu, YiSheng Huang, KeYi Xing, Kamel Barkaoui, Murat Uzam, Hans-Michael Hanisch, GuanJun Liu, Maria Pia Fanti, Shouguang Wang, Khalgui Mohamed, Olfa Mosbahi, and ZhiMing Zhang for their constructive criticisms and insights. The authors are also thankful to Professor Jing Shao, Xi’an JiaoTong University, for her assistance in literature search.

Appendix

Basics of Petri Nets
A Petri net [165] is a four-tuple $N = (P, T, F, W)$ where $P$ and $T$ are finite and nonempty sets. $P$ is the set of places and $T$ is the set of transitions with $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$. $F \subseteq (P \times T) \cup (T \times P)$ is called flow relation of the net, represented by arcs with arrows from places to transitions or from transitions to places. $W : (P \times T) \cup (T \times P) \to \mathbb{N}$ is a mapping that assigns a weight to an arc: $W(f) > 0$ if $f \in F$ and $W(f) = 0$ otherwise, where $\mathbb{N} = \{0, 1, 2, \ldots\}$. $N = (P, T, F, W)$ is called an ordinary net, denoted as $N = (P, T, F)$, if $\forall f \in F, W(f) = 1$. A marking $M$ of $N$ is a mapping from $P$ to $\mathbb{N}$. $(N, M_0)$
is a P-invariant of \((P, T, F, W, M_0)\). For economy of space, we use \(\sum_{p \in P} M(p)p\) to denote vector \(M\).

Let \(x \in P \cup T\) be a node of \(N = (P, T, F, W)\). The preset of \(x\) is defined as \(\ast x = \{ y \in P \cup T \mid (y, x) \in F \}\). While the postset of \(x\) is defined as \(x^* = \{ y \in P \cup T \mid (x, y) \in F \}\).

This notation can be extended to a set of nodes as follows: given \(X \subseteq P \cup T\), \(\ast X = \cup_{x \in X} \ast x\), and \(X^* = \cup_{x \in X} X^*\). Note that \(\ast X\) is the preset of \(\ast X\) and \(X^*\) is the postset of \(X^*\).

Given place \(p\), we denote \(\max \{ W(p, t) \mid t \in p^* \}\) by \(\max_{p^*}\).

A transition \(t \in T\) is enabled at a marking \(M\) if \(\forall p \in \ast t\) \(M(p) \geq W(p, t)\); this fact is denoted as \(M[t]\); firing \(t\) yields a new marking \(M'\) such that \(\forall p \in P\), \(M'(p) = M(p) - W(p, t) + W(t, p)\); it is denoted as \(M[t]M'\). Marking \(M'\) is said to be reachable from \(M\) if there exists a sequence of transitions \(\sigma = t_0t_1\ldots t_n\) and markings \(M_1, M_2, \ldots, M_n\) such that \(M[t_0]\rangle M_1 \rangle \ldots \rangle M_n \rangle M'\) holds. The set of markings reachable from \(M\) in \(N\) is denoted as \(R(N, M)\).

A net is pure (self-loop free) if \(\exists x, y \in P \cup T\), \(\exists (x, y) \in F\) \& \((y, x) \in F\). A pure net \(N = (P, T, F, W)\) can alternatively be represented by its incidence matrix \([N]\), where \([N]\) is a \(|P| \times |T|\) integer matrix with \([N](p, t) = W(t, p) - W(p, t)\), where \(|P|\) \& \(|T|\) means the cardinality of set \(P\) \& \(T\). For a place \(p\) in \(N\)’s incidence vector is denoted as \([N]p\).

A transition \(t \in T\) is live under \(M_0\) if \(\forall M \in R(N, M_0), \exists M' \in R(N, M), \exists M''[t]\). \(N\) is deadlock-free if \(\forall M \in R(N, M_0), \exists t \in T\), \(\exists M[t]\). \(N, M_0\) is live if \(\forall t \in T, t\) is live under \(M_0\). \(N, M_0\) is bounded if \(\exists k \in \mathbb{N}\), \(\forall M \in R(N, M_0), \forall p \in P\), \(\exists M(p) \leq k\). \(N, M_0\) is quasi-live if \(\forall t \in T, \exists M \in R(N, M_0), M[t]\) holds. \(N, M_0\) is said to be reversible, if for each marking \(M \in R(N, M_0), M_0\) is reachable from \(M\).

A P-vector is a column vector \(I : P \rightarrow \mathbb{Z}\) indexed by \(P\) and a T-vector is a column vector \(J : T \rightarrow \mathbb{Z}\) indexed by \(T\), where \(\mathbb{Z}\) is the set of integers. A P(T)-vector \(I(J)\) is denoted by \(\sum_{p \in P} I(p)p\ (\sum_{t \in T} J(t)t)\) for economy of space. We denote column vectors where every entry equals 0(1) by 0(1). \(I^T\) \& \([N]^T\) are the transposed versions of a vector \(I\) and matrix \([N]\), respectively. P-vector \(I\) is a P-invariant (place invariant) if \(I \neq 0\) and \(I^T[N] = 0^T\). P-invariant \(I\) is said to be a P-semiflow if no element of \(I\) is negative. \(|I| = \{ p \in P \mid I(p) > 0 \}\) is called the support of \(I\). \(|I|^+ = \{ p \mid I(p) > 0 \}\) denotes the positive support of P-invariant \(I\), whereas \(|I|^- = \{ p \mid I(p) < 0 \}\) denotes the negative support of \(I\). An invariant is said to be minimal when its support is not a strict superset of the support of any other, and the greatest common divisor of its elements is one. If \(I\) is a P-invariant of \((N, M_0)\) then \(\forall M \in R(N, M_0), I^T[M] = I^T M_0\).

Let \(N = (P, T, F, W)\) be a Petri net with \(P_X \subseteq P\) and \(T_X \subseteq T\). \(N|_{P_X \cup T_X} = (P_X, T_X, F_X, W_X)\) is called a subnet generated by \(P_X \cup T_X\). \(F_X = F \cap ([P_X \times T_X] \cup \{T_X \times P_X\})\) \& \(\forall f \in F_X, W_X(f) = W(f)\).

Let \(S\) be a non-empty subset of places. \(S \subseteq P\) is a siphon (trap) if \(\ast S \subseteq S^*\ (S^* \subseteq \ast S)\). A marked trap can never be emptied. A siphon is said to be minimal if it contains no other siphons as its proper subset. A minimal siphon is strict if it contains no trap. A siphon is said to be controlled if it can never be emptied. A siphon \(S\) is said to be invariant-controlled by P-invariant \(I\) if \(I^T M_0 > 0\) and \(|I|^+ \subseteq S\).

Let \((N, M_0)\) be a generalized Petri net and \(S\) be a siphon of \(N\). \(S\) is max-controlled if \(\forall M \in R(N, M_0), \exists p \in S, M(p) \geq \max_{p^*} W(p, t)\). \((N, M_0)\) is said to satisfy the controlled-siphon property (cs-property) if each minimal siphon of \(N\) is max-controlled. If \((N, M_0)\) is live, then it satisfies the cs-property. If \((N, M_0)\) satisfies the cs-property, then it is deadlock-free [10].
References


