Improved Design of Concurrent Synchronization System Controller Based on Petri Net

Xu Yang[1,*], Shaocui Guo[2], Dongming Xiang[3], Yuxin Yang[4] and Yijun Chen[5,*]

[1]*Department of Computer Science, Tongji University, Shanghai 201804, China*
[2]*Open Education College, Yantai Vocational College, Yantai 264670, China*
[3]*The School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China*
[4]*The Experimental High School Affiliated to Shenzhen University, Shenzhen 518083, China*
[5]*National Maglev Transportation Engineering R&D Center, Tongji University, Shanghai 201804, China*
E-mail: 1610482@tongji.edu.cn; gscytvc@126.com; flysky_xdm@163.com; yy2964@tc.columbia.edu; chenyijun@tongji.edu.cn
[*]*Corresponding Author*

Received 07 April 2023; Accepted 08 December 2023;
Publication 03 February 2024

Abstract

At present, the advancement of science and technology has contributed to the emergence of various network parallel environments, web service concurrency environments and massively parallel processors. In order to abstract some practical systems into concurrent system models, it is necessary to conduct model analysis, and to perform deadlock detection of the concurrent system. Despite prior studies on the control problem of the synchronous concurrent system, there remains room for improvement in terms of workload, time, and efficiency. Some very practical methods are explored by introducing locking and unlocking to synchronize the concurrent system, and the algorithm that involves the specific regulation is provided. The Petri net model of the controller is developed to provide an effective method applicable to find the deadlock and to prevent and eliminate the deadlock for synchronous concurrent systems.

Keywords: Synchronization concurrent system, Petri net, deadlock detection, deadlock controller, concurrency, deadlock elimination, Petri net with data, data consistent.

1 Introduction

Preventing and eliminating deadlocks [1, 2] play an essential role in ensuring the security of various concurrent systems. Removing the deadlock of circular possession of resources is to either deprive one party of resources to the other party, or to regain the resources. One method undermines fairness, while the other may cause the deadlock to recur.

Petri net is one of the research hotspots in the field of computer science, providing a solution to modelling and the control of discrete event systems (DES). Applicable in discrete event systems, Petri net also provides a mathematical framework to remove the deadlock from various concurrency systems. It is effective in preventing the deadlock in automated manufacturing systems (AMS).

For concurrent systems, there are two ways to conduct deadlock analysis. One is to find the deadlock marking using a reachability graph. The other is that the incidence matrix solution is used to judge which transitions are causal, concurrent and conflicting relationships for subsequent deadlock analysis. As a regulation or requirement for the system behavior, specification is introduced into Petri net according to the reachability graph. It can include a set of constraints applied to limit the operation of the system, thus ensuring the correctness and reliability of the system.

The synchronization of the concurrent events [8, 9] is to execute a pair of critical areas protected by lock synchronization sequentially twice. Through deadlock avoidance [10, 11] and deadlock elimination, the deadlock is removed from the root. Thus, further deadlock is prevented. To remove the deadlock of cycle possession of resources is either to deprive one party of resources to the other party, or to regain the resources. One method undermines the fairness, while the other may cause the same deadlock to recur.

S^3PR net is premised on a control/supervisory place. In a Petri net with resources, the deadlock can be removed by introducing the control place that is a special type of place. It plays a role in limiting the allocation and use of resources, thus preventing the deadlock.

An incidence matrix is based on place-invariant. In the Petri net, the incidence matrix and P-invariants are combined to remove the deadlock. As a property of calculating the state of the Petri net, the P-invariant is used to describe the pattern of the number of markers in the place over transitions. By analyzing the incidence matrix and P-invariants, the position of deadlock can be traced and it can be eliminated. Siphon and trap is the basic structure of Petri net. If the siphon lacks token in a marking, there remains no token in successor marking. That is to say, a siphon loses tokens only. The trap contains at least one token in a marking, and then it is marked in each post marking. That is to say, the trap can gain tokens only. Deadlocks can be prevented by using siphons and traps to analyze the liveness of the Petri net.

One cause of deadlock is the uneven distribution of resources. For example, there are excess resources on one assembly line, while there are no resources on the other assembly line. The solution to this is very simple; that is, the excess resources are transferred to the assembly line lacking in resources. Another cause of the deadlock is as follows. When a resource r1 is occupied by one thread thread1, an application is made for a resource r2 that has been preoccupied by another thread thread2. Thus, thread1 and thread2 are made to wait for each other.

The solution in the current practice is detailed as follows. The resources r1 and r2 are treated as a whole to apply for r1 and r2 simultaneously. After the resource is used, the resources r1 and r 2 are released at the same time. In this paper, a new solution is proposed. According to this solution, an inhibitor arc is used to inhibit other transitions for occupying r2, which prevents resource r2 from being occupied by thread1.

In Section 2, the basic process of eliminating deadlocks is followed to obtain the marking by building the reachability tree or graph. As for deadlock marking, the cause of deadlock is analyzed. Based on the marking deadlock, the specification basic algorithm is applied to remove the deadlock. In Section 3, different targeted solutions are provided for different deadlock cases. In Section 3.1, deadlock occurs due to limited resources. For this reason, the resources are used in this paper to eliminate the deadlock. In Section 3.2, when a thread occupies the resources and the resources occupied by other threads are requested, the threads wait to be occupied, and the deadlock is eliminated by increasing the inhibitor arc. In Section 3.3, the conversion of resource is performed to remove the deadlock with one pipeline awaiting resource and another pipeline with excessive resource. In Section 3.4, a more common way is adopted to solve the deadlock, which is to introduce the control/supervision place. Through the incidence matrix, the initial token of the place is analyzed. With the corresponding arc added, the deadlock is eliminated. In Section 3.5, the Petri net with data operation is proposed to formally describe the multi-threaded concurrent system. In this paper, it is concluded that data reading is inconsistent. Therefore, lock and unlock mutually exclusive resources need to be added in the code to ensure the consistency of data reading and writing. In Section 4, a summary of this paper is presented.

2 Controller Synthesis Algorithm

Definition 1 (Petri net). N = (P,T,F) where P is a set of places, T is a set of transitions, F is a flow relation $\mathbf{F} \subseteq (\mathbf{P} \times \mathbf{T}) \cup (\mathbf{T} \times \mathbf{P})$, and P $\cap$ T $= \emptyset$. Figure 1 shows a net, where a circle represents a place (P), a bar or box represents a transition (T), and the arc between the box and circle represents the flow (F) relations.

In Figure 1 the set $\{s_1, s_2\}$ is a siphon; the set $\{s_3, s_4\}$ is a trap, denoted as s_2 and s_1 respectively. Therefore,

$$^\bullet\{s_1, s_2\} \subseteq \{s_1, s_2\}^\bullet.$$

This property is referred to as siphons. Therefore,

$$\{s_3, s_4\}^\bullet \subseteq {}^\bullet\{s_3, s_4\}.$$

This property is referred to as traps.

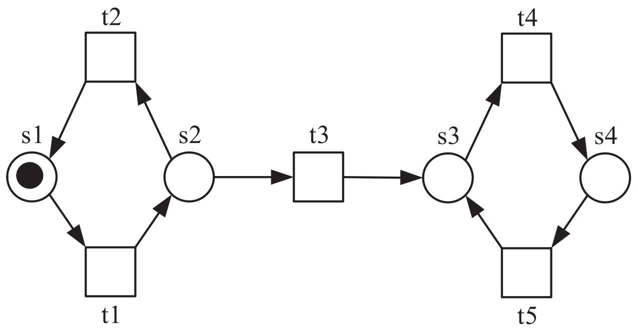


Figure 1 Siphon and trap.

According to the definition of siphon and trap, if the siphon is out, it cannot gain entry again. As a trap, once it enters a trap, it will never get out again. The precursor of Siphon is included in the successor, and the successor of the trap is included in the precursor.

Definition 2 (deadlock). Let Pn = (P,T,F, M0) be a Petri net and Me be the end marking. M $\in$ R(Pn,M0) is a deadlock w.r.t Md (deadlock marking for short) if $\forall \mathrm{t} \in \mathrm{T}, \wedge \mathrm{M}[\mathrm{t} > \wedge \mathrm{Md} \neq \mathrm{Me}$.

Definition 3 (livelock). Let Pn = (P,T,F,M0) be a Petri net and Me be the end marking. M $\in$ R(Pn,M0) is a livelock w.r.t Mv (deadlock marking for short) if $\mathrm{Mv} \in \mathrm{R(Pn, M0)}, \ni \mathrm{t} \in \mathrm{T}, \mathrm{Mv}[\mathrm{t} > \wedge \mathrm{Mv} \neq \mathrm{Me}$.

It is necessary to apply control on the deadlock of the system. In this system, the controller is introduced to remove the deadlock integrating the control specification. The specific process is as follows:

Step 1: From the Petri net model of the original system, its reachable marking diagram (RMG) is generated.
Step 2: From the original system reachable marking diagram RMG and the control specification (Spec) [2–5], the reachable state graph of the target system is generated.
Step 3: From the target system reachable marking diagram, the reachable marking diagram RMG (C) [6–8] of the controller is generated.
Step 4: From the reachable marking diagram RMG (C) of the controller, the Petri net model (C) of the controller is generated.

2.1 Algorithm 1

Enter: $\sum = (P, T, F, M_0)$.

Output: $\mathrm{RMG}(\sum) = (V, E; f)$.

Step 1: Sets $V \leftarrow \{M_0\}, E \leftarrow \emptyset$, the initial value and marks M_0, it as "new".

Step 2: If there is no "new" node in V, the construction process ends, otherwise go to Step 3.

Step 3: Select $M \in V$, a "new" node and do the following:

Step 3.1: $\forall t \in T$, if, $M[t >$ do the following work:

Step 3.1.1: Find out M', which makes $M[t >$.

Step 3.1.2: If $M' \in V$, then set $E \leftarrow E \cup \{(M, M)\}, f((M, M')) \leftarrow t$, turn to Step 3.2, otherwise revert to Step 3.1.1.

Step 3.1.3: If $\exists M'' \in V$, $\forall p \in P$, and $M''(p) \leq M'(p)$, all the p.s.t $M''(p) < M'(p)$ that $M'(p) \leftarrow \infty$ are satisfied.

Step 3.1.4: Set $V \leftarrow V \cup \{M'\}$, $E \leftarrow E \cup \{(M, M')\}$, $f((M, M')) \leftarrow t$, mark “new”, and revert to Step 3.1.

Step 3.2: Cross out M, the “new” and revert to Step 2.

According to the literature [9], the finite termination and correctness of the Petri net reachability tree generation algorithm are performed. Combining the same identification nodes in the reachability tree, the reachability graph algorithm [10–12] ensures the limited termination and correctness of the reachability graph generation algorithm.

2.2 Algorithm 2

Input: RMG $(\sum)$, Spec//Spec is a set of linear inequalities about the identification//

Output: $\mathrm{RMG}(\sum \Theta \mathrm{C})$.

Step 1: According to the actual requirements, T_c determines the controlled transition set and T_u determines the uncontrolled transition set. Thus, $T_c \cap T_u = \emptyset$, and $T_c \cup T_u \neq \emptyset$.

Step 2: $\forall t_c \in T_c$ is correct, for which p_c place is controlled by the transition t_c. Thus, the control place P_c is collected.

Step 3: Increase the length $|P|$ of the identification vector of any node $M \in V$ in the RMG $(\sum)$ to $|P| + |P_c|$.

Step 4: According to the requirements of the specification, solve P_c, which is the value of the part of the sub-vector, and fill in the value of the corresponding sub-vector in the processed RMG $(\sum)$ for compliance with the requirements of the specification. Thus, $\mathrm{RMG}(\sum \Theta \mathrm{C})$ is obtained.

Spec is a set of linear inequalities about the identity, involving the conservation equations, each of which contains one of the identification variables. Since the node identification in the RMG $(\sum)$ is deterministic, the partial vector P_c can be solved. Due to the constraints of the specification, the deadlock in the RMG $(\sum)$ is prevented. This $\mathrm{RMG}(\sum \Theta \mathrm{C})$ contains no deadlocks.

2.3 Algorithm 3

Enter: $\mathrm{RMG}(\sum$ and $C) = (V, E, f)$.

Output: $\mathrm{RMG(C)} = (VC, EC, fC)$.

Step 1: $VC = \{\Gamma_{P-P_c}(M) | (\forall M \in V) \wedge (\Gamma_{P \to P_c}(M))\}$, PC is the projection vector.

Step 2: $\forall M'$, $M'' \in V$, if $(M', M'') \in E$, and $f((M', M'')) = t$, then set $EC \leftarrow EC \cup \{(\Gamma_{P\to P_c}(M), \Gamma_{P\to P_c}(M''))\}$, $f_c(\Gamma_{P-P_c}(M), \Gamma_{P\to P_c}(M'')) \leftarrow t$.

Since $\sum \Theta \mathrm{C}$ it is actually $\sum$ synchronized with C, for $\forall t \in T_{\sum \Theta \mathrm{C}}$, there are cases as follows.

Case 1: If $t \in T_{\sum} - T_C$, then $M'[t > M''$, iff $\Gamma_{P\to P\sum}(M')[t > \Gamma_{P\to P\sum}(M'')$ and $\Gamma_{P\to P_c}(M') = \Gamma_{P\to P_C}(M'')$.

Case 2: If $t \in T_{\sum} - T_C$, then $M'[t > M''$, iff $\Gamma_{P\to P_c}(M')$ $[t > \Gamma_{P\to P_C}(M'')$ and $\Gamma_{P\to P\sum}(M') = \Gamma_{P\to P\sum}(M'')$.

Case 3: If $t \in T_{\sum} \cap T_C$, then $M'[t > M''$, iff $\Gamma_{P\to P_c}(M'')[t > \Gamma_{P\to P\sum}(M'')$ and $\Gamma_{P\to P_c}(M')[t > \Gamma_{P\to P\sum}(M'')$.

In this way, $\mathrm{RMG(C)} = (V_c, E_c, f_c)$, which is actually $\mathrm{RMG}(\sum \Theta \mathrm{C})$ projected on C.

The algorithm for generating a reachable tree is Algorithm 4.

2.4 Algorithm 4

Input: RMG (C).

Output: $\mathrm{C} = (Pc, Tc, Fc, Mc_0)$.

Step 1: Set $F_c \leftarrow \emptyset$, mark $M_{c0} = \Phi$, and push M_{c0} to stack.

Step 2: If the stack is not empty, do Step 3, otherwise it ends.

Step 3: Top node M_c that is not marked, then proceed to Step 4 or Step 5 otherwise.

Step 4: $\Delta MC = MC - \mathrm{stack(top)}$, remember $f(\mathrm{stack(top)}, MC) = t$, and proceed to Step 4.1.

Step 4.1: $\forall p_c \in P_c$, if $\Delta MC(pC) > 0$; that set $FC \to FC \cup \{t, pC\}$, if $\Delta M_c(p_c) < 0$, that set $F_c \leftarrow F_c \cup \{(p_c, t)\}$.

Step 4.2: Push P_c and M_c into stack, to run Step 3.

Step 5: Pop the stack and run Step 2.

According to the state equation of the net theory [10–12], it is known that $\mathrm{M_c}{-}\mathrm{stack(top)} = A^T t$, where A^T represents the transferred array of the correlation/incidence matrix [9] of the net C. $t = f((\mathrm{stack(top)}, M_c))$. Thus, from every two nodes of the RMG (C) and their associated edges, the previous equation of state is used to obtain the net (P_c, T_c, F_c). The structure of C about the initial identification of C, denoted as M_{c0}, is obtained by identifying the initial node of the RMG (C). Then, the net $\mathrm{C} = (P_c, T_c, F_c, M_{c0})$ is obtained.

3 Handling of the Deadlock Phenomenon

Deadlocks are one of the anomalies that are likely to occur in concurrent systems [13–15]. If they are not eliminated, they would lead to the failure of the whole system. In this section, it is analyzed how the deadlock can be prevented.

One of the contributors to deadlock is the uneven distribution of resources, such as the circumstance that there are excess resources on one assembly line but no resources on the other assembly line. A simple solution to this problem is to direct the excess resources to the assembly line with a lack of resources.

In Figure 2(a1), $\sum_1$ is a Petri net. In Figure 2(a2), the Petri net suffers deadlock. Figure 2(b) shows the reachable marking diagram RMG $(\sum_1)$. As can be seen clearly from Figure 2(c), there are two deadlock states, of which one (02000) is caused by t1 and the other (00200) is caused by t2. Figure 2(d) shows the reachability graph in which the deadlock is controlled.

In order to eliminate the deadlock, the control places s_1 and s_2 are introduced, and they agree on the control specification.

$$\text{Spec} = \{M(p_2) < 2, M(p_3) < 2, M(p_2) + M(s_1) = 1,\\ M(p_3) + M(s_2) = 1 | M \in R(M_0)\}.$$

Since $M_0(p_3) = M_0(p_2) = 0$, $M_0(s_2) = M_0(s_1) = 1$, and the RMG $(\sum_1)$ evolves into RMG $(\sum_1 \Theta C_1)$, as shown in Figure 2(c). According to Algorithms 3 and 4, respectively, the RMG, including (C_1) (Figure 2(b)) and C_1 (Figure 2(c)), are obtained. What C_1 is obtained is the Petri net of the controller model that is sought. The structure C_1 should be noted.

3.1 Method 1

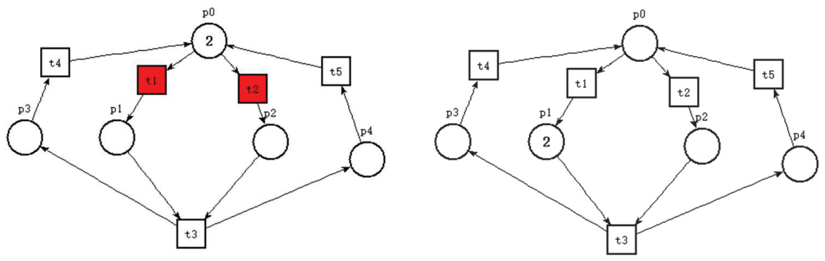
Introducing resources to eliminate deadlock.

The lack of resources is one of the main reasons for the deadlock, as shown in Figure 2. If a token is added into p_0, it is easy to eliminate the deadlock, as shown in Figure 3.

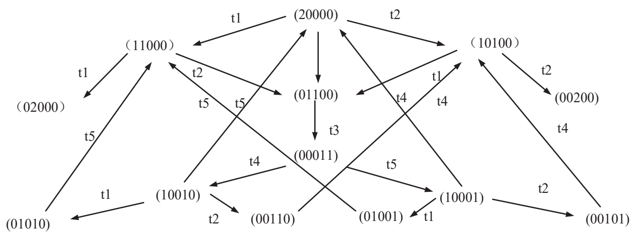
3.2 Method 2

Introducing the inhibitor arc to avoid deadlock.

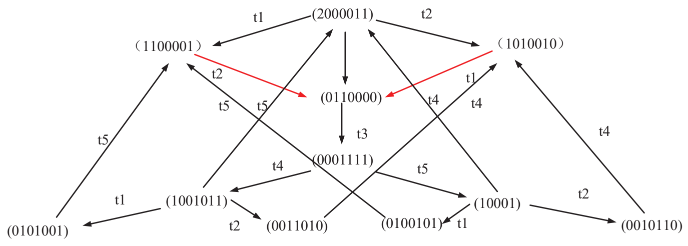
Another reason for the deadlock is that resource r1 is occupied by one thread thread1 and then resource r2 is requested that has been occupied by another thread thread2. Thus, thread1 and thread2 wait for each other.



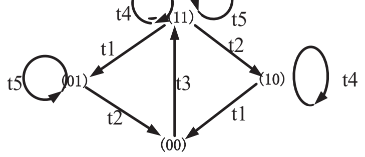
(a1) A Petri net. (a2) The emergence of the deadlock.



(b) RMG $\left(\Sigma_1\right)$.



(c) RMG $\left(\Sigma\Theta C\right)$.



(d) RMG $\left(C\right)$.

Figure 2 Petri net and the RMG, control place/transition, and the controlled RMG.

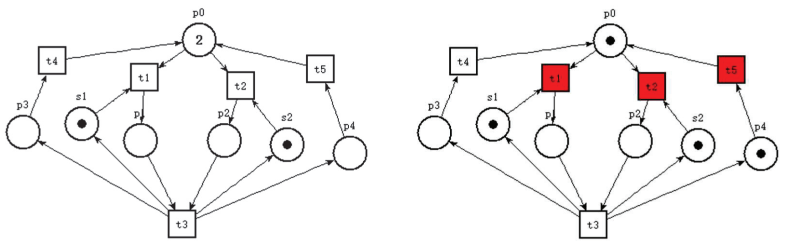


Figure 3 Introducing resources to eliminate deadlock.

The solution is as follows. Resources r1 and r2 are taken as a whole to apply for r1 and r2 at the same time. After the resource is used, resources r1 and r2 are released at the same time. In this paper, a new solution is proposed by means of the inhibitor arc. The resource r1 inhibits other transitions to occupy r2, thus occupying resource r2.

Definition 4 (inhibitor net). An inhibitor net is a tuple InN = (P,T,F,I,m0), where P is a set of places, T is a set of transitions, $P \cap T = \Phi$, $F \subseteq (P \times T) \cup (T \times P)$ is a flow relation, $I \subseteq P \times T$ is a set of inhibitor arcs, and $m0 \subseteq P$ is the initial marking. An inhibitor arc $(p,e) \in I$ means that e is enabled only if p is unmarked. In graphs, the inhibitor arc (p, e) is indicated by an edge with a small circle at the end, and any set of places $m \subseteq P$ is referred to as a marking.

When there is already a token in p1 or p2, which means t1 or t2 has occurred once in Figure 2, the inhibitor recurs, as shown in Figures 4(a) and 4(b). The reachability graph is shown in Figure 4(c).

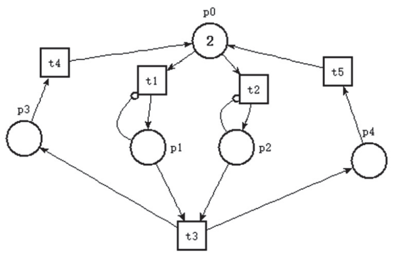
3.3 Method 3

Add the mutually conversion transition.

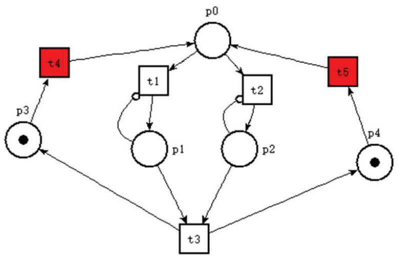
Adding mutual conversion transitions means that two tokens are positioned in one place, after a transition into the corresponding place synchronized with the place shown in Figure 5. The reachability graph is presented in Figure 6 where no deadlock nodes exist.

3.4 Removing the Deadlock: Implementation Usage of the Resource Controller

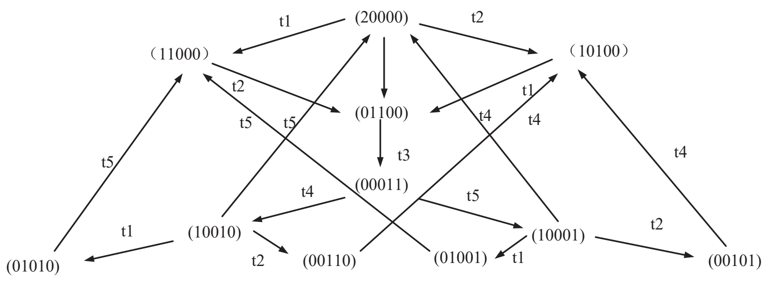
In this paper, the red transition represents the enabling transition. That is to say, the transition in red can be fired. In Figure 7, there is no transition that can be fired, which means no red transition is present.



(a) Introducing an inhibitor arc.

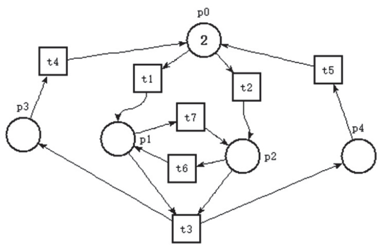


(b) Running without deadlock.

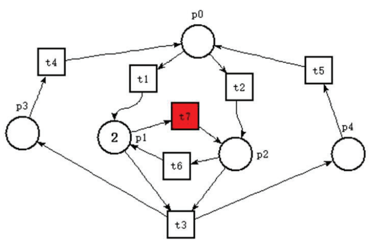


(c) Inhibitor arc improved RMG$\left(\sum_1\right)$.

Figure 4 Using the inhibitor arc to avoid deadlock.



(a) The occurrence of deadlock on the net.



(b) Introducing transition to eliminate deadlock.

Figure 5 Introducing the mutually converted transition.

According to the business flow net in Figure 7(b) (deadlock occurs), deadlock occurs due to the competition for shared resources. Add Figure 8 for access control of resources to achieve Figure 9(a) for dead free. With the concept of atomicity introduced, t1 or t5 or p9 and p10 show simultaneity and avoid deadlock in Figure 9(b). In order to ensure the safety and stability of the system, control place is introduced, as shown in Figure 10(a)(b).

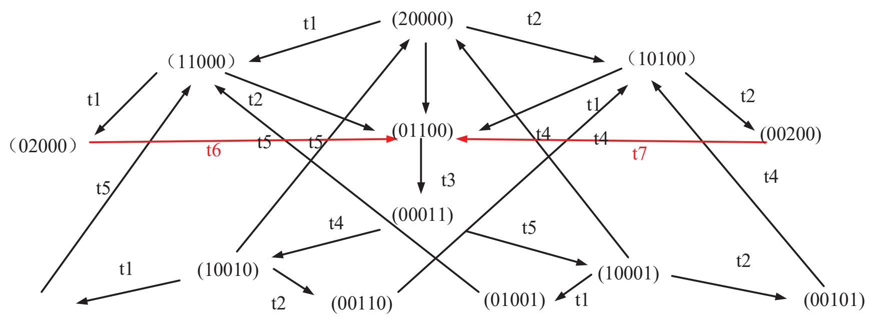
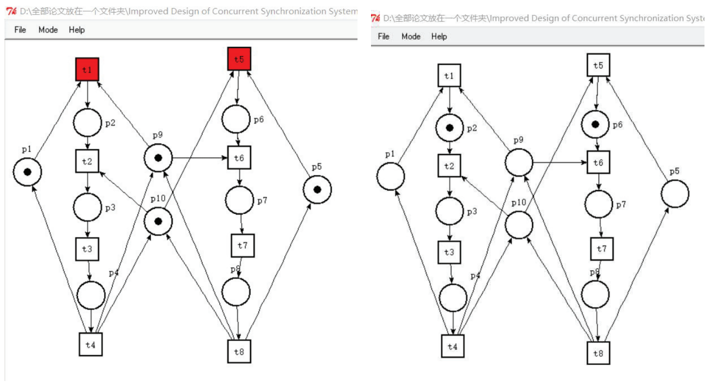


Figure 6 RMG ($\sum_1$) for converted transition improvement.


D:\全部论文放在一个文件夹\Improved Design of Concurrent Synchronization System
File
Mode
Help
t1
t2
t3
t4
t5
t6
t7
t8
p1
p2
p3
p4
p5
p6
p7
p8
p9
p10

(a) Exemplary net. **(b)** Deadlock.

Figure 7 Deadlock occurs to the net.

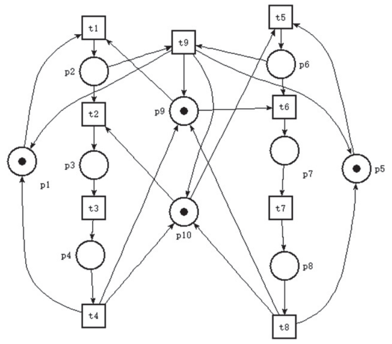


Figure 8 One solution of introducing t9 to automatically release the deadlock when it occurs.

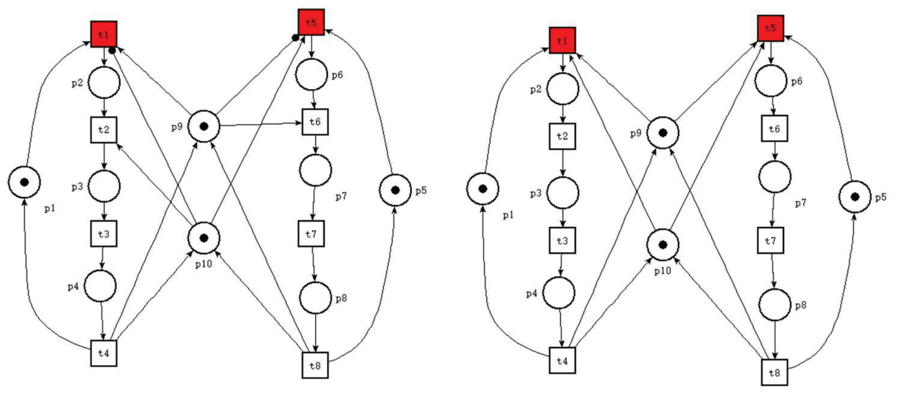


Figure 9 (a) Introducing reading arc to avoid deadlock. (b) Introducing reading arc between p10→t1 and p9→t5.

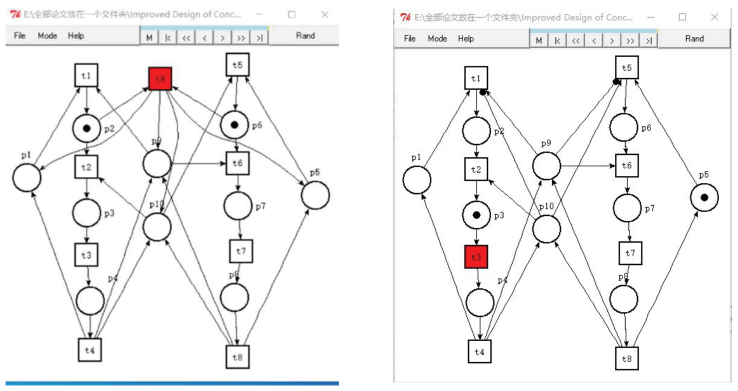


Figure 10 (a) Introducing control place to release the deadlock and (b) read arc to release the deadlock.

3.5 Removing Inconsistent Data: Implementation Usage of the Lock and Unlock Mutex

Definition 5 (Petri network with data, DPN). A 9-tuple $\Sigma = $ (Pc, Tc, Fc, Pd, Tr, Tw, Fr, Fwr, Fww) is called a Petri net with data, if it satisfies the following conditions:

(1) Pc is the control library, Pd is the database, and they meet $P_c \cap Pd = \emptyset$;
(2) Tc is a collection of control changes;
(3) Fc is a control arc, including $Tc \times Pc \cup Pc \times Tc$;

```
      init x:=0;y:=0
thread 1      ||   thread 2
x:=2;         ||   x:=1;
r1:=x;        ||   r2:=y;
If(r1!=2)     ||   If(r2!=0)
then          ||   then
y:=1;         ||   x:=3;
```

Figure 11 Concurrency code.

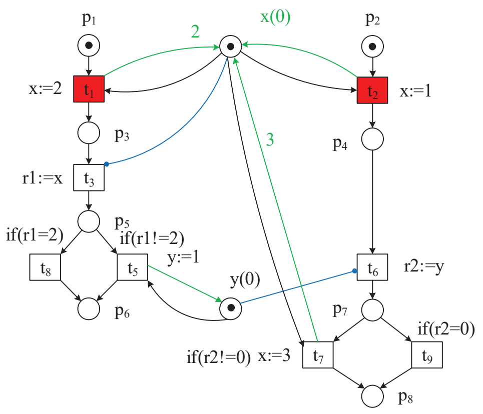


Figure 12 A DPN description of the concurrent code (Figure 11).

(4) Fd reading and writing data arc, $\{Tr, Tw\} \times \mathrm{Pd} \cup \mathrm{Pd} \times \{\mathrm{Tr}, \mathrm{Tw}\}$, where Fd includes reading data (Fr), writing data (Fwr, Fw) arc, i.e. Fr:Pd $\times$ Tr and Fw (Fwr, Fww) i e. Fw:(Pd $\times$ Tw, Tw $\times$ Pd).
The operation on the data is as follows: $F_d \rightarrow P_d$ and $P_d \rightarrow F_d$, and $P_d = \{[D_1, val_1], [D_2, val_2]\ldots, [D_n, Val_n]\}$ is a finite set of shared data elements D, where val_i is the value of D_i.

(5) C: m $(P_c \cup P_d)$ $\{0,1,2\ldots\}$, indicating the configuration or state, where D = (m, d), m is marking, d is the data value, while C_0 and D_0 are the initial configuration.

For the pd $\in$ Pd of the data in DPN, the function getValue (pd) is applied to obtain its value. For convenience, a set of implementation data operation functions are proposed as well. Read: getValue (pd), Write: setValue (pd) = D. The database is described in the form of (Key, Value), and it is also simply recorded as x (value), where x is the shared variable name.

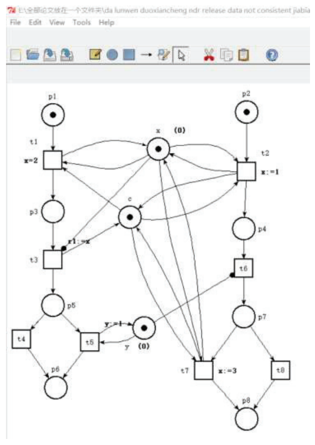


Figure 13 The introduced control place c.

```
        init x:=0;y:=0
  thread 1     ||  thread 2
  Lock(c)      ||    x:=1;
   x:=2;       ||    r2:=y;
   r1:=x;      ||  If(r2!=0)
 Unlock(c)     ||    then
 If(r1!=2)     ||    x:=3;
then y:=1;     ||
```

Figure 14 Locking/unlocking to the concurrency code.

Figure 11 shows a concurrent code of two threads used to access the shared variables x and y. An inconsistency of data reading occurs in Figure 12 where description is made using Petri net with data: r1 may be 1 or r1 is 2 or 3. As shown in Figure 13, the control place is added to synchronously control the access of x. As shown in Figure 14, locking and unlocking are performed for the synchronous coding of x.

4 Conclusion

As a solution to concurrent system modeling, Petri net is widely used for deadlock control. Currently, it is common to study the method of Petri net-based synchronous concurrency control. In this paper, the deadlock caused by synchronous concurrency control is explored. The method proposed in this study and the research result provide some guidance. Petri net provides an effective solution to model detection and construction in concurrent systems. It is worth exploring the methods used to detect dead-lock and live-lock in concurrent systems, and the elimination of dead-lock and live-lock is achievable. In this paper, the existence of dead-lock and live-lock is determined by calculating the reachability graph and using the correlation matrix. With the assistance of a reading arc and inhibitor arc, the Petri net with data reading and writing place can eliminate deadlock, achieve live-lock and enable the control of inconsistency in data reading and writing, thus ensuring the correctness of the concurrency system.

Through the removal of deadlocks, the marking is obtained by constructing a reachability tree or graph. For the purpose of deadlock marking, the cause of the deadlock is analyzed. With the deadlock eliminated from the graph, the specification basic algorithm is applied to eliminate the deadlock. Different targeted solutions are provided for different cases of deadlock. Due to the deadlock caused by the limited availability of resources, there is an increase in the resources needed to eliminate the deadlock. When a thread occupies the resources and applies for the resources occupied by other threads, the threads wait to be occupied, and the deadlock is eliminated by increasing the inhibitor arc. The conversion of resource is performed to remove the deadlock with one pipeline waiting resource and another pipeline with excessive resource. The more common solution to the deadlock is to increase the control/supervision place. Through the correlation matrix, the initial token of the place is analyzed and the corresponding arc is introduced, thus eliminating the deadlock. The Petri net with data operation is proposed to make the formal description of the multi-threaded concurrent system. It is concluded that the data reading is inconsistent. Therefore, the locking and unlocking of mutually exclusive resources are required for coding to ensure the consistency of data reading and writing.

Furthermore, the corresponding controller is integrated through the introduction time Petri model and control specification. The running trace is reduced by analyzing the running trace of the concurrent system on the Petri net with data reading and writing.

Funding

This research is funded by the National Natural Science Foundation of China (62172166), the Shanghai Maglev and Rail Transit Collaborative Innovation Center, and the Science Research Project of Shanghai Science and Technology Committee (Grant No. 18DZ1205803). The Huang Yanpei Education Foundation (ZJS2022YB363).

Conflicts of Interest

The authors declare that there are no conflicts of interest to report in this manuscript.

References

[1] D. Xiang, S. Lin, X. Wang and G. Liu, "Checking Missing-Data Errors in Cyber-Physical Systems Based on the Merged Process of Petri Nets," in IEEE Transactions on Industrial Informatics, vol. 19, no. 3, pp. 3047–3056, March 2023, doi: 10.1109/TII.2022.3181669.

[2] L. He and G. Liu, "Prioritized Time-Point-Interval Petri Nets Modeling Multiprocessor Real-Time Systems and TCTL_x," in IEEE Transactions on Industrial Informatics, vol. 19, no. 8, pp. 8784–8794, Aug. 2023, doi: 10.1109/TII.2022.3222342.

[3] F. Zhao, D. Xiang, G. Liu and C. Jiang, "A New Method for Measuring the Behavioral Consistency Degree of WF-Net Systems," in IEEE Transactions on Computational Social Systems, vol. 9, no. 2, pp. 480–493, April 2022, doi: 10.1109/TCSS.2021.3099475.

[4] L. Qi, Y. Su, M. Zhou and A. Abusorrah, "A State-Equation-Based Backward Approach to a Legal Firing Sequence Existence Problem in Petri Nets," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 53, no. 8, pp. 4968–4979, Aug. 2023, doi: 10.1109/TSMC.2023.3241101.

[5] G. Liu, W. Reisig, C. Jiang and M. Zhou, "A Branching-Process-Based Method to Check Soundness of Workflow Systems," in IEEE Access, vol. 4, pp. 4104–4118, 2016, doi: 10.1109/ACCESS.2016.2597061.

[6] Z. Zhang, G. Liu, K. Barkaoui and Z. Li, "Adaptive Deadlock Control for a Class of Petri Nets with Unreliable Resources," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 52, no. 5, pp. 3113–3125, May 2022, doi: 10.1109/TSMC.2021.3062469.

[7] D. Xiang, G. Liu, C. Yan and C. Jiang, "A Guard-Driven Analysis Approach of Workflow Net with Data," in IEEE Transactions on Services Computing, vol. 14, no. 6, pp. 1650–1661, 1 Nov.–Dec. 2021, doi: 10.1109/TSC.2019.2899086.

[8] Yang, Xu. 'Performance Analysis of Petri Net Based on Moment Generating Function'. 1 Jan. 2023: 1131–1139. https://content.iospress.com/articles/journal-of-intelligent-and-fuzzy-systems/ifs231137.

[9] Y. Huang, T. Wang, Z. Yin, E. Mercer and B. Ogles, "Improving the Efficiency of Deadlock Detection in MPI Programs Through Trace Compression," in IEEE Transactions on Parallel and Distributed Systems, vol. 34, no. 1, pp. 400–415, 1 Jan. 2023, doi: 10.1109/TPDS.2022.3218346.

[10] Liu, G. (2020). PSPACE-Completeness of the Soundness Problem of Safe Asymmetric-Choice Workflow Nets. In: Janicki, R., Sidorova, N., Chatain, T. (eds) Application and Theory of Petri Nets and Concurrency. PETRI NETS 2020. Lecture Notes in Computer Science(), vol. 12152. Springer, Cham. https://doi.org/10.1007/978-3-030-51831-8_10.

[11] Yang, X., Ye, C., Chen, Y. Depth-First Net Unfoldings and Equivalent Reduction. Symmetry 2023, 15, 1775. https://doi.org/10.3390/sym15091775.

[12] Xu Yang and Chen Ye. Analysis of Concurrent Systems Based on Interval Order MVLSC Volume 42, Number 1–3 (2024).

[13] S. Wang, X. Guo, O. Karoui, M. Zhou, D. You and A. Abusorrah, "A Refined Siphon-Based Deadlock Prevention Policy for a Class of Petri Nets," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 53, no. 1, pp. 191–203, Jan. 2023, doi: 10.1109/TSMC.2022.3174421.

[14] G. Liu, W. Reisig, C. Jiang and M. Zhou, "A Branching-Process-Based Method to Check Soundness of Workflow Systems," in IEEE Access, vol. 4, pp. 4104–4118, 2016, doi: 10.1109/ACCESS.2016.2597061.

[15] Chao DY, Yu TH. "MLR: A new concept to launch a partial deadlock avoidance policy for k-th order system of Petri Nets", Industrial Electronics Society IECON 2015 – 41st Annual Conference of the IEEE, pp. 003148–003152, 2015.

[16] Yu TH. "Parameterized of Control Related States of Gen-Right k-th order system of Petri nets based on proof by model of Gen-Left", Industrial Electronics Society IECON 2016 – 42nd Annual Conference of the IEEE, pp. 276–281, 2016.

[17] Chao DY, Yu TH, Chen TY. "Computation of Control Related States of Middle k-th Order System (with a Nonsharing Resource Place) of Petri Nets", Computer Consumer and Control (IS3C) 2014 International Symposium on, pp. 244–247, 2014.

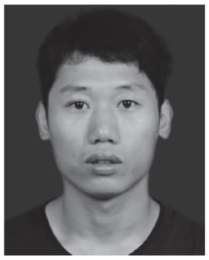
[18] W. Luan, L. Qi, Z. Zhao, J. Liu and Y. Du, "Logic Petri Net Synthesis for Cooperative Systems," in IEEE Access, vol. 7, pp. 161937–161948, 2019, doi: 10.1109/ACCESS.2019.2950971.

[19] Watt C, Pulte C, Podkopaev A, et al. 2020. Repairing and mechanising the JavaScript relaxed memory model. In Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 2020). Association for Computing Machinery, New York, NY, USA, 346–361. doi: 10.1145/3385412.3385973.

Biographies

Xu YANG (1977'02–) male, doctoral candidate, He graduated from Qufu Normal University with a bachelor's degree in 1999, and graduated from Tianjin Polytechnic University with a master's degree in 2006. Now he is studying in Tongji University for a doctor's degree. member of CCF. Research fields: formal methods, concurrency theory.

Shaocui Guo (1980'11–), female, associate professor. In 2004, he graduated from Qufu Normal University with a bachelor's degree in computer science. After graduation, She taught in the Department of Computer Science of Open education college, Yantai vocational college. His main research interests are machine learning (deep learning), big data analysis and its application in the field of industrial intelligence.



Dongming Xiang received the Ph.D. degree in Computer Science and Technology from Tongji University, Shanghai, China, in 2018. He is currently an associate professor with the Department of Computer Science and Technology, Zhejiang Sci-Tech University. He has authored over 25+ papers including TII, TSC, TCSS, JAS, and ICPADS. His research interests include model checking, Petri net, formal methods, business process management, and service computing.

Yuxin Yang (1997’07–), female. She graduated from Hong Kong Baptist University with a bachelor’s degree in Translation and Interpretation in 2019, and graduated from Columbia University with a master’s degree in TESOL in 2021. Now she is working as an English teacher in The Experimental High School Affiliated to Shenzhen University.

Yijun Chen (1971’05–) male, senior engineer, member of CCF. Research fields: formal methods, runtime verification.