Computing and Informatics, Vol. 39, 2020, 174–192, doi: 10.31577/cai_2020_1-2_174

A METHOD FOR LEARNING A PETRI NET MODEL BASED ON REGION THEORY

Jiao Li, Ru Yang, Zhijun Ding

The MOE Key Laboratory of Embedded System and Service Computation
Tongji University
Shanghai, 201804, China
e-mail: {li_jiao, yangru}@tongji.edu.cn, zhijun_ding@outlook.com

Meiqin Pan*

School of Business and Management
Shanghai International Studies University
Shanghai, 200083, China
e-mail: panmqin@sina.com

Abstract. The deployment of robots in real life applications is growing. For better control and analysis of robots, modeling and learning are the hot topics in the field. This paper proposes a method for learning a Petri net model from the limited attempts of robots. The method can supplement the information getting from robot system and then derive an accurate Petri net based on region theory accordingly. We take the building block world as an example to illustrate the presented method and prove the rationality of the method by two theorems. Moreover, the method described in this paper has been implemented by a program and tested on a set of examples. The results of experiments show that our algorithm is feasible and effective.

Keywords: Petri net, robot model, robot learning, region theory, Petri net synthesis

Mathematics Subject Classification 2010: 93A30

* Corresponding author

1 INTRODUCTION

Robotics systems and techniques which appeared during the recent years have achieved astonishing development, and they not only facilitate humans' lives but also replace humans' work in some difficult situations. With the continuous expansion of the field of robot applications, higher requirements are needed for the safety, correctness and reliability of robots. In order to more effectively control a robot system and verify the system properties, it is necessary to build a model for the robot system.

In the field of robotics, many researchers have worked on modeling. Desai [1] proposed a new modeling method that can control multiple teams of mobile robots navigating in a terrain with obstacles, while maintaining a desired formation and changing formations when required. Wieber et al. [2] studied the modeling method of legged robots, and used the model to generate and control the dynamic motions, as well as analyze the stability of the robot. However, in some situations, deliberative planning or pre-programming to achieve tasks will not be always possible. Hence, there is a growing research interest in imbuing robots not only with the capability of perception and planning but also of learning [3]. According to current state of the art of robot learning, most of the successful results presented in the literature are applied by machine learning. There are many different implementation methods, such as reinforcement learning, artificial neural network and evolutionary techniques [4, 5, 6]. However, there are few studies on the modeling and learning of robots via formal methods.

Petri nets (PNs) [7] are a powerful formal modeling tool, which have advantages in the intuitiveness of its graphical modeling and the rigor of its analytical theory. Especially, they have the ability to describe the complex logical relationships between systems or process activities, such as concurrency, competition, synchronization, and order. Moreover, there are many PN modeling tools [8] which provide the functions of establishing, modifying, storing, and dynamic simulation that can be used to analyze and valid the properties of the PNs. Therefore, they have been widely applied in various fields, including the field of robotics. Lima et al. [9] introduced distinct Petri net types to model robotic tasks from different views of the robotic task model. Ziparo et al. [10] presented a language (Petri Net Plans) based on PNs, which allows for intuitive and effective robot and multi-robot behavior design. Chao et al. [11] developed a system for multimodal collaboration based on a timed Petri net representation, and implemented action interruptions in reciprocal interaction within the system. In these applications, PNs were mostly created manually rather than automatically. Chang et al. [12] proposed a learning method that automatically creates PNs from observation of human demonstrations to model the underlying structure of tasks. Different from most of the existing methods, this work enables PNs to be created automatically. But the operation sequences of imitation need to be designed artificially. It is our hope that the robot can learn a PN model in a limited number of task-oriented attempts without manual planning.

The attempts of the robot can be regarded as the system behavior. There are two major approaches to obtain a PN model from the information of system behavior. One approach is process mining technology [13, 14]. Although a PN model can always be gained by a process mining technology, the obtained model is not necessarily consistent with the actual model. The other way to transform the behaviors to a structure description model (PN) is related to the PN synthesis problem [15], whose method is mainly based on the region theory [16]. The original goal of PN synthesis is to construct an elementary PN according to a given transition system and test whether the reachability graph of the PN is isomorphic to the transition system. If it is isomorphic, such a PN is constructed. Nowadays, there are many extensions in this field, such as changing the transition system into formal languages and execution traces or changing the target from elementary PNs to Place/Transition nets. Several tools for synthesis have already existed, like petrify [17], genet [18], synet [19] and apt [20]. By this method, we can convert the effective attempts of a robot into an accurate PN model that describes the operation process of the robot in a compact form. At the same time, we can learn some rules from the limited attempts to enrich the known information so as to obtain a more complete model.

This paper proposes a PN model generation algorithm based on the region theory and gives two theorems as well as proofs to guarantee the rationality of the method. Also, to illustrate how to obtain a more complete PN model automatically within the robot's limited attempts, the problem of the robot in the building block world is taken as an example. The main contributions are as follows:

1. An automatic generation and learning scheme of robot model from the robot's limited attempts based on PN is given. It provides a new idea for robot model learning by using PN.
2. Two theorems are proved, which are the theoretical basis of this paper. One theorem guarantees the accuracy of the model generated from a transition system and the other reveals the rationality of adding information so that any transition system getting from a robot system can generate a PN model.
3. A PN model generation algorithm that provides an operational method for the scheme is recommended. It transforms the behaviors of the robot into a PN model and rationally supplements the information based on the region theory and two above theorems. By this means, the purpose of automatic model generation and learning is achieved.

The next section describes a classical problem in building block world and expounds the problem to be solved in this paper from the perspective of robot learning and control. Section 3 briefly reviews the basics of PN and some definitions related to the proposed approach. After proving two theorems, the PN model generation algorithm is given. In order to confirm the feasibility of the proposed method, some examples are shown in Section 4. Finally, conclusions and outlooks are presented in Section 5.

2 MOTIVATING EXAMPLE

In some situations, due to the uncertainty of the environment or difficulty of comprehensive analysis, it is hard to build accurate models manually. Taking the classical problem of building block world as an example, it is difficult to consider all the operation sequences of the robot and obtain a complete model manually. Thus, it requires a method which can generate a model automatically as well as learn some information from the existing information. There is no doubt that we can control robots better with a more complete model.

A building block world scene is as follows: a number of blocks on a table are placed and a robot is asked to change the initial state of the blocks to the target state. The robot has a mechanical arm (hand) and just can perform the specified actions which are shown in Table 1.

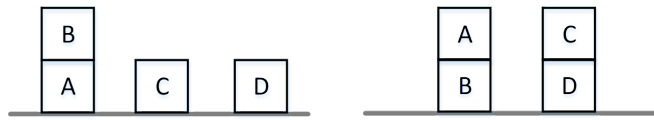
Action	**Explanation**	**Precondition**
unstack(A, B)	Pick up building block A from building block B.	Building block A is stacked on building block B; there is no other building block on building block A; the robot's hand is empty.
putdown(A)	Place building block A on the table.	Building block A is in robot's hand.
pickup(A)	Pick up building block A from the table.	Building block A is on the table; there is no other building block on building block A; the robot's hand is empty.
stack(A, B)	Place building block A on building block B.	There is no other building block on building block B; building block A is in robot's hand.

Table 1. The actions of robot

We define that if the robot executes the action "unstack(A, B)", it must put the building block A on the table before taking another action. That is to say, it is not allowed to place the building block A on another building block after executing the action "unstack(A, B)". So we can combine the action "unstack(A, B)" and the action "putdown(A)" into one operation, that is, "unstack(A, B)-putdown(A)" is an atomic operation. In addition, we suppose the robot can only pick up one building block at a time which means robot needs to put down block in hand before executing another action, so "pickup(A)-stack(A, B)" and "pickup(A)-putdown(A)" are also atomic operations. Because the atomic operation "pickup(A)-putdown(A)" doesn't change the state of the building block, we consider to ignore it. In summary, the operations that the robot can perform are "unstack(A, B)-putdown(A)" and "pickup(A)-stack(A, B)". For ease of writing, we will record the first operation as USPD(A, B) and the second one as PUS(A, B).

For states of this scene, we only consider the upper and lower relative positions of the blocks and the position of the blocks and the table, regardless of the left and

right relative positions of the blocks. For example, Figure 1 shows the initial state and target state of the building blocks that we assume. As in the target state, we only require the building block A is on the building block B, the building block C is on the building block D and the building blocks B and D are on the table, without concerning about the left and right relative positions of the building block AB and the building block CD. If multiple operations can be executed in a situation, robot can perform an operation at will until the state of all blocks is consistent with the target state.



a) b)

Figure 1. a) Initial state and b) target state of the building blocks

Then, based on the above scene assumptions, how can robots learn a PN model by limited attempts without human intervention? This question is answered in this paper.

3 METHODS

In this section, we first give the basic concepts of PN and PN synthesis, which are derived from [15, 21, 22, 23]. Next, we present two theorems which are the theoretical foundation of the proposed method. Finally, based on the definitions and theorems, we put forward the algorithm for generating PN models.

3.1 Preliminaries

Definition 1. A net is a quad $N = (P, T, F, W)$, where P is a finite set of places, T is a finite set of transitions such that $P \neq \varnothing$, $T \neq \varnothing$, $P \cap T = \varnothing$, $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation, and W is a weight function such that $W(x, y) \in \mathbb{N}^+$ (here $\mathbb{N}^+ = \{1, 2, 3, \dots\}$) if $(x, y) \in F$ and $W(x, y) = 0$ if $(x, y) \notin F$.

The marking of a net is a function $M : P \to \mathbb{N}$ (here $\mathbb{N} = \{0, 1, 2, 3, \dots\}$). It is represented by a multiset expression or a $|P|$-vector $(M(p_1), \dots, M(p_{|P|}))^T$, where $M(p)$ is the number of tokens in place $p \in P$. A PN $PN = (N, M_0)$ is a net N with an initial marking M_0.

A transition $t \in T$ is said to be enabled at marking M, which is denoted as $M[t >$, if $\forall p \in P$, $M(p) \geq W(p, t)$. Firing an enabled transition t results in changing M into M', represented by $M[t > M'$, where $\forall p \in P, M'(p) = M(p) - W(p, t) + W(t, p)$. A sequence of transitions $\sigma = t_1 t_2 \dots t_k$ is a firing sequence if there exists a sequence of markings such that $M[t_1 > M_1[t_2 > \dots M_{k-1}[k > M_k$, it

can be written as $M[\sigma > M_k$, and M_k is said to be reachable from M by firing σ. The reachability set $R(M)$ is a set of all markings reachable from M.

The reachability graph of PN is a directed graph with the set of vertices $R(M_0)$, and arcs $\{(M, t, M') | M, M' \in R(M_0) \wedge M[t > M'\}$.

Definition 2. A place $p \in P$ is said to be bounded or K-bounded if $\forall M \in R(M_0)$, $M(p) \leq K$, where $K \in \mathbb{N}^+$ (here $\mathbb{N}^+ = \{1, 2, 3, \dots\}$). A PN PN is said to be bounded if its every place is bounded. If $K(PN) = \max\{K(p) | p \in P\} = 1$, PN is a safe PN.

Here, we only consider the PN with an arc weight of 1, so the weight function W of the PN can be omitted, represented by $PN = (P, T, F, M_0)$. Unless otherwise stated, the PNs referred in this paper are all safe PNs.

Definition 3. A transition system (S, E, Δ) consists of a set of states S, a set of events E, and a set of transitions $\Delta \subset S \times E \times S$. An initialized transition system $TS = (S, E, \Delta, S_0)$ consists of a transition system (S, E, Δ) and an initial state $s_0 \in S$.

An event e is enabled in a state s, denoted by $s \xrightarrow{e}$, if there is a state s' such that $(s, e, s') \in \Delta$. This situation is written as $s \xrightarrow{e} s'$ and means that state s' is reachable from state s through the execution of event e. The definitions of enabledness and of the reachability relation are extended as usual to event sequences (or directed paths) $\sigma \in E^* : s \xrightarrow{\varepsilon}$ and $s \xrightarrow{\varepsilon} s'$ are always true; $s \xrightarrow{\sigma e}$ ($s \xrightarrow{\sigma e} s'$) iff there is a state s'' with $s \xrightarrow{\sigma} s''$ and $s'' \xrightarrow{e}$ ($s'' \xrightarrow{e} s'$, respectively). A state s' is reachable from a state s if there is an event sequence σ such that $s \xrightarrow{\sigma} s'$. A state s' is reachable if it is reachable from state s_0. By $s^{\rightarrow}$, we denote the set of states reachable from state s.

Definition 4. An initialized transition system $TS = (S, E, \Delta, s_0)$ is called finite if S and E (hence also Δ) are finite sets. It is deterministic if for any reachable state s, s', s'' and event e, $s \xrightarrow{e} s'$ and $s \xrightarrow{e} s''$ implies $s' = s''$ and it is totally reachable if $S = s_0^{\rightarrow}$ and $\forall e \in E : \exists s \in s_0^{\rightarrow} : s \xrightarrow{e}$.

A transition system characterizes the migration process of the system states, which can be either artificially designed or actually obtained. It should be noted that the transition systems involved in this paper are all gained by robot's actual attempts.

Definition 5. Two $TS_1 = (S_1, E, \Delta_1, s_{01})$ and $TS_2 = (S_2, E, \Delta_2, s_{02})$ over the same set of evens E are isomorphic if there is a bijection ζ: $S_1 \to S_2$ with $\zeta(s_{01}) = s_{02}$ and $(s, t, s') \in \Delta_1 \Leftrightarrow (\zeta(s), t, \zeta(s')) \in \Delta_2$, for all $s, s' \in S_1$.

The reachability graph of a PN PN can be seen as an initialized transition system. If there is a PN PN whose reachability graph is isomorphic to a given initialized transition system TS, then we will say that PN solves TS [23].

Definition 6. A region of an initialized transition system $TS = (S, E, \Delta, s_0)$ is a triple $(\mathbb{R}, \mathbb{B}, \mathbb{F}) \in N^S \times N^E \times N^E$ such that the following holds:

$$\forall s \xrightarrow{e} s' \in \Delta : \mathbb{R}(s) \geqslant \mathbb{B}(e) \wedge \mathbb{R}(s') = \mathbb{R}(s) - \mathbb{B}(e) + \mathbb{F}(e).$$

In the above formula, the first condition states that no transition in the initialized transition system may be prevented, and the second condition enforces consistency between $\mathbb{R}$, $\mathbb{B}$, and $\mathbb{F}$. Intuitively, this describes a possible place in a PN generating TS where $\mathbb{B}(e)$ and $\mathbb{F}(e)$ describe the number of tokens consumed and produced, respectively, by a transition $e \in E$, and $\mathbb{R}(s)$ is the number of tokens on this place in state $s \in S$ [23].

For every region $(\mathbb{R}, \mathbb{B}, \mathbb{F})$ if $s \xrightarrow{\sigma} s'$ for some s, $s' \in S$ and $\sigma = e_{a1}e_{a2}\ldots e_{ak} \in E^*$, then $\mathbb{R}(s') = \mathbb{R}(s) + \sum_{i=1}^{k}(\mathbb{F}(e_{ai}) - \mathbb{B}(e_{ai}))$. Since we are assuming that the TS is totally reachable, $\mathbb{R}$ is thus fully determined by $\mathbb{R}(s_0)$ via $\mathbb{R}(s) = \mathbb{R}(s_0) + \sum_{i=1}^{n} \psi_s(e_i) \cdot (\mathbb{F}(e_i) - \mathbb{B}(e_i))$, where $\psi_s(e_i)$ is the number of times that e_i occurs in σ when $s_0 \xrightarrow{\sigma} s$. We identify a region $\rho = (\mathbb{R}, \mathbb{B}, \mathbb{F})$ with a vector $\rho \in N^{1+2n}$:

$$\rho = (\rho_0, \ldots, \rho_{2n}) = (\mathbb{R}(s_0), \mathbb{B}(e_1), \ldots, \mathbb{B}(e_n), \mathbb{F}(e_1), \ldots, \mathbb{F}(e_n)).$$

The function that reconstructs the value $\mathbb{R}(s)$ for a state $s \in S$ from such a vector is given by $\text{tokens}(\rho, s) = \rho_0 + \sum_{i=1}^{n} \psi_s(e_i) \cdot (\rho_{n+i} - \rho_i)$.

Definition 7. For a region set R of an initialized transition system $TS = (S, E, \Delta, s_0)$, the corresponding PN $PN = (P, T, F, M_0)$ has $P = R$, $T = E$ and for each $\rho = (\mathbb{R}_\rho, \mathbb{B}_\rho, \mathbb{F}_\rho) \in R$ defines $F(\rho, e) = \mathbb{B}_\rho(e)$, $F(e, \rho) = \mathbb{F}_\rho(e)$ and $M_0(\rho) = \mathbb{R}(s_0)$. If the reachability graph of the corresponding PN is isomorphic to the TS, i.e., TS is isomorphic to the reachability graph of the net system synthesized from R, we will say that the region set R solves TS.

For example, a region set $R = \{(1,0,1,0,1,0,0,0), (1,1,0,0,1,1,1,0)\}$ can be found in the transition system shown in Figure 2 a). For each region in R, we can define a place and the flow relationship between the place and transitions in the PN model. As shown in Figure 2 b), the region $\rho_1 = (1,0,1,0,1,0,0,0)$ corresponds to place p_1, and the region $\rho_2 = (1,1,0,0,1,1,1,0)$ corresponds to place p_2. Moreover, it can be seen that the reachability graph of PN_1 shown in Figure 2 c) is isomorphic to transition system TS_1, that is, the region set R solves TS_1.

Remark 1. It is a hope that the number of places in the PN is as small as possible, so it leads to the emergence of sink transitions. The sink transitions will not affect the normal behavior of the PN.

Definition 8. A state separation problem $\text{SSP}(s, s')$ is a set of two states $\{s, s'\} \subseteq S$ with $s \neq s'$ that must be distinguishable and it is solved by a region ρ with $\mathbb{R}_\rho(s) \neq \mathbb{R}_\rho(s')$. The corresponding predicate is $\text{SSP}(\rho, s, s') := (\text{tokens}(\rho, s) \neq \text{tokens}(\rho, s'))$. A counterexample is given in the Figure 3 a) which shows an initialized transition

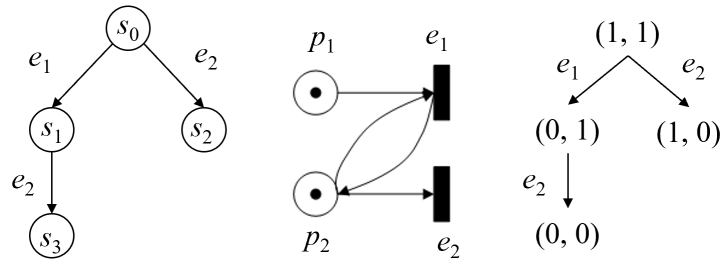


Figure 2. a) A transition system TS_1, b) the corresponding PN PN_1, and c) the reachability graph of the PN_1

system in which states s_3 and s_4 cannot be separated by any region. If all the state separation problems of a TS can be solved, we call this TS satisfies the state separation property.

An event/state separation problem $\mathrm{ESSP}(s, e)$ is a pair $(s, e) \in S \times E$ with $\neg(s \xrightarrow{e})$. This problem is solved by a region $(\mathbb{R}_\rho, \mathbb{B}_\rho, \mathbb{F}_\rho)$ iff $\mathbb{R}_\rho(s) < \mathbb{B}_\rho(e)$, which means that event e is prevented in state s. This is expressed by the predicate $\mathrm{ESSP}(\rho, s, e_i) := (\mathrm{tokens}(\rho, s) < \rho_i)$. One of its counterexamples is shown in the Figure 3 b). It demonstrates that event e_3 cannot be separated from state s_1 by any region in the initialized transition system. If all the event/state separation problems of a TS can be solved, we call this TS satisfies the event/state separation property.

The set of all separation problems of TS is called SP. For readability, given any kind of separation problem $pr \in \mathrm{SP}$, we define $\mathrm{SP}(\rho, pr)$:

$$\mathrm{SP}(\rho, pr) := \begin{cases} \mathrm{SSP}(\rho, s, s') = (\mathrm{tokens}(\rho, s) \neq tokens(\rho, s')), & \text{if } pr = \mathrm{SSP}(s, s'), \\ \mathrm{ESSP}(\rho, s, e_i) = (\mathrm{tokens}(\rho, s) < r_i), & \text{if } pr = \mathrm{ESSP}(s, e_i). \end{cases}$$

3.2 Relevant Theorems

According to the above definitions, we present the following two theorems which provide theoretical support for the subsequent algorithm. The first theorem states that the model generated by the algorithm is accurate. The second guarantees that any transition system can generate a PN model by supplementing the transitions if it failed, and the supplement of the information is reasonable.

Theorem 1. If there is a region set R of the initialized transition system $TS = (S, E, \Delta, s_0)$ that can solve all separation problems SP in the TS and satisfies $\forall p \in R$, $\mathrm{tokens}(\rho, s) \leq 1$, where $s \in S$, then the corresponding PN is safe and its reachability graph is isomorphic to the TS.

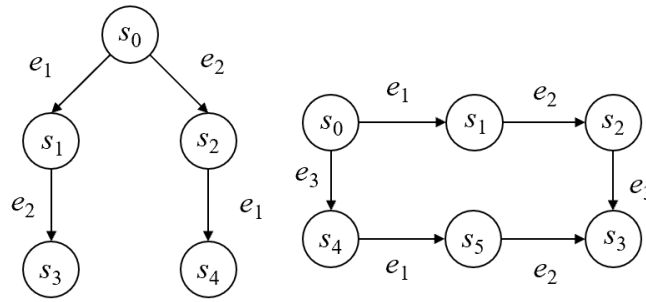


Figure 3. a) A transition system where state separation fails, and b) a transition system where event/state separation fails

Proof. Let $SN(TS)$ be the PN obtained by $TS = (S, E, \Delta, s_0)$, and its reachability graph is $RG(SN(TS))$, which is denoted as $TS_2 = (S_2, E_2, \Delta_2, s_0 2)$. Assuming that TS_2 is not isomorphic to TS, then either the event sets of TS_2 and TS are different or there is no bijection $\zeta : S \to S_2$ with $\zeta(s_0) = s_0 2$ and $(s, t, s') \in \Delta \Leftrightarrow (\zeta(s), t, \zeta(s')) \in \Delta_2$, for all $s, s' \in S$. In view of synthesis method, the event sets of TS_2 and TS are the same. Then we consider the second assumption, that is, $\exists s, s' \in TS_2$, $\zeta(s_0) \neq s_0 2$ or $(s, t, s') \in \Delta_1 \Leftrightarrow (\zeta(s), t, \zeta(s')) \in \Delta_2$ is not satisfied. Because the region set R can solve all the separation problems SP in TS, by Definition 6, the region in R considers all the constraints of $s \xrightarrow{e} s' \in \Delta$ and $\neg(s \xrightarrow{e})$ for all $s, s' \in S$ and $e \in E$, there must exist a bijection $\zeta : S \to S_2$ with $\zeta(s_0) = s_0 2$ and $(s, t, s') \in \Delta \Leftrightarrow (\zeta(s), t, \zeta(s')) \in \Delta_2$, for all $s, s' \in S$, which opposites to the assumption. In addition, $\forall s \in S$, $\rho \in R$, $\text{tokens}(\rho, s) \leq 1$ conforms to the definition of safe PN. Hence, the proof is complete. □

This theorem is proposed based on the summary of the existing conclusions and here we give its proof. For a more detailed introduction of region theory, please refer to [16].

Theorem 2. Let $TS = (S, E, \Delta, s_0)$ be an initialized transition system and it satisfies the state separation property. For any event/state separation problem $\text{ESSP}(s, e)$ in the TS, if there is no region to solve the problem, then $s \xrightarrow{e}$ or $s \xrightarrow{e} s'$ (s' is a new state) can be added to the TS, and this supplement is reasonable.

Proof. For an event/state separation problem $\text{ESSP}(s, e)$ in TS, i.e. $\neg(s \xrightarrow{e})$, if the problem can be solved by a region ρ, then ρ satisfies $\mathbb{R}_\rho(s) < \mathbb{B}_\rho(e)$. But such region does not exist, that is to say, all regions ρ satisfy $\mathbb{R}_\rho(s) \geq \mathbb{B}_\rho(e)$, which means $s \xrightarrow{e}$ by Definition 6. Since $\mathbb{R}_\rho(s') = \mathbb{R}(s) - \mathbb{B}(e) + \mathbb{F}(e)$, state s' can be calculated. If s' does not exist in the state set S, it is necessary to add it into S. It can be seen from the above analysis that the event set is not changed and the supplemental state can

be obtained by occurring the event actually, what complies with the rules of the system. Hence, the supplement is reasonable. □

In Theorem 2, we prove that it is reasonable to add some transitions to the TS when facing the failure of PN generation. Corresponding to the real scene, the other information gained according to the known one by the robot is consistent with the actual. Obviously, this is a manifestation of learning.

3.3 Generation Algorithm for Petri Net Model

As stated above, we can construct a PN according to an initialized transition system TS based on the region theory. If the regions of the TS satisfy the certain conditions, a PN whose reachability graph is isomorphic to the initial transition can be generated. In a robot scene, an initialized transition system can be obtained by the records of execution sequences and state changes. Then, we can use it to produce a PN model automatically. If it failed to generate a PN model, this work gains some information from the known one and adds them to the initial transition system, which shows the robot has the ability to learn. Here we introduce the PN model generation algorithm.

Algorithm 1 Petri Net Model Generation Algorithm

Input: an initialized transition system $TS = (S, E, \Delta, s_0)$
Output: a PN Model PN
1) Let the region set Π and unresolved separation problem set Ξ be $\varnothing$(empty);
2) For each separation problem $pr \in$ SP in TS, do
3) If find a region ρ can solve pr and $\forall s \in S$, tokens$(\rho, s) \leq 1$, then
4) Put ρ into the set Π;
5) Else
6) Put pr into the set Ξ;
7) End if
8) End for
9) If unresolved separation problem set Ξ is not empty, then
10) For each separation problem ESSP(s_i, e_i) in Ξ, do
11) Calculate state s_i' which is reachable from s_i through the execution of e_i
12) If state s_i' is not in state set S, then
13) Add state s_i' to state set S;
14) End if
15) Add transition $s_i \xrightarrow{e_i} s_i'$ to transition set Δ;
16) End for
17) Return step 1; // Repeat steps, where TS has been changed.
18) Else
19) Synthesize a PN model PN by region set Π according to Definition 7;
20) End if
21) Output PN model PN;
End

In Algorithm 1, we first calculate solutions to all separation problems in TS by using a general PN synthesis algorithm [23] (steps 3–4). Then for the purpose of adding information to construct a PN whose reachability graph is isomorphic to the TS, we put the current unsolvable separation problem(s) into the set Ξ (step 6). If the set Ξ is empty, the PN model can be synthesized by the region set Π according to the Definition 7 (step 19); otherwise, some information needs to be added. As for an actual system, we require that any two states of the system should be distinguished, so the TS which comes from the reality satisfies the state separation property, that is, there are only event/state separation problems in the unresolved separation problem set Ξ. Thus, we can add some arcs (or arcs with states) by performing the steps 10 to 16. It can be seen from Theorem 2, the added information is reasonable. After that, return to step 1 and re-solve problems in the new TS until the region set of TS satisfies the conditions. Finally, the PN model PN can be output (step 21).

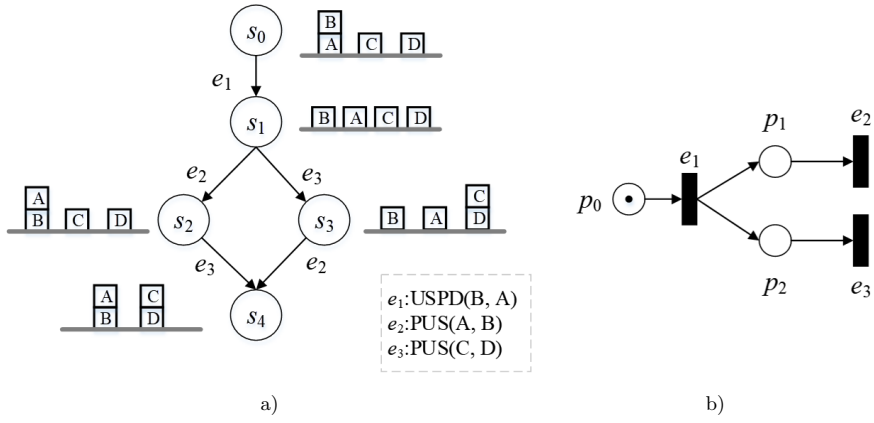


Figure 4. a) The initialized transition system TS_2, and b) the corresponding PN PN_2

Here we use the example shown in Figure 1 to explain the algorithm. It is assumed that a part of the initialized transition system TS can be obtained by the robot's autonomous attempts shown in Figure 4 a), where a state is represented by s_i $(0 \leq i \leq 4)$ and an event is represented by e_i $(1 \leq i \leq 2)$. For convenience, we show the position of building blocks next to the state and list the operations in the dashed box. Taking the TS as an input of Algorithm 1, the output can be obtained as shown in Figure 4 b). In Algorithm 1, the first operation is to solve each separation problem $pr \in \mathrm{SP}$ in TS. For example, in state s_2, the event e_2 is not enabled, so $pr = \mathrm{ESSP}(s_2, e_2)$ is an event/state separation problem. Because state s_2 is reached from state s_0 via event sequence e_1e_2, we can obtain $\mathrm{SP}(\rho, pr) = (\mathrm{tokens}(\rho, s_2) < \rho_2) = (\rho_0 + 1 \cdot (\rho_4 - \rho_1) + 1 \cdot (\rho_5 - \rho_2) + 0 \cdot (\rho_6 - \rho_3) < \rho_2)$ according to the Definition 6 and the Definition 8. In addition, the following inequalities can

be produced owing to the region constraints:

$$\begin{aligned}
&\rho_i \geq 0 \ (0 \leq i \leq 6)\\
&\wedge \rho_1 \leq \text{tokens}(\rho, s_0) = \rho_0\\
&\wedge \rho_2 \leq \text{tokens}(\rho, s_1) = \rho_0 + (\rho_4 - \rho_1)\\
&\wedge \rho_3 \leq \text{tokens}(\rho, s_1) = \rho_0 + (\rho_4 - \rho_1)\\
&\wedge \rho_3 \leq \text{tokens}(\rho, s_2) = \rho_0 + (\rho_4 - \rho_1) + (\rho_5 - \rho_2).
\end{aligned}$$

Considering the target PN is a safe PN, the constraint $\text{tokens}(\rho, s_i) \leq 1$ $(0 \leq i \leq 4)$ should also be satisfied. We can compute that the vector $\rho = (0, 0, 1, 0, 1, 0, 0)$ is a possible solution of the above constraints. In this example, there is no unsolvable separation problem, so the PN can be obtained according to the Definition 7 as shown in Figure 4 b), where the vector $\rho = (0, 0, 1, 0, 1, 0, 0)$ is corresponding to place p_1 in the PN.

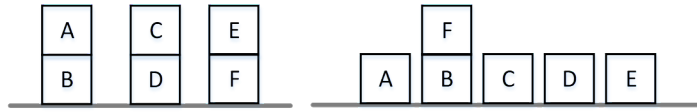
4 SIMULATION AND EXPERIMENTS

In Section 3, we give the algorithm of Petri net model generation and prove the corresponding theorems to ensure the rationality of the algorithm. In order to exhibit the effectiveness of the method better, in this section, we achieve a program to simulate the scene introduced in Section 2 and implement the algorithm. After experiments and comparisons, it is indicated that the method is reasonable and effective.

The program is coded in Java and requires input of the initial state and the target state of the building blocks. In the program, when a path from the initial state to the target is found, the Algorithm 1 is called to generate the PN model. In the process of simulation, there may be cases where the known information is insufficient and the PN cannot be obtained. At this time, some information needs to be added according to the unresolved separation problem set, that is, steps 9 to 17 of Algorithm 1 will be executed. Then an accurate PN model which is more complete can be obtained.

In the beginning, we introduce another example, as shown in Figure 5. Figures 5 a) and 5 b) show the initial state of the building block and the target one. When the initialized transition system TS_1 is produced by robot's attempts, as shown in Figure 6 a), it fails to generate a PN model. Therefore, the information can be supplemented according to the unresolved event/state separation problems. That is to say, the transitions s0 $\xrightarrow{e2}$ s10, s10 $\xrightarrow{e0}$ s5 and s10 $\xrightarrow{e1}$ s9 (s10 is a new state) should be added to TS_1 what results in TS_2, as shown in Figure 6 b). The initial state of the TS is marked in red labeled with s0 and the target state is marked in green labeled with s4. Besides, we mark the position of the building blocks in the state. For instance, "A-B" in s0 means that the building block A is on the building block B, and "@" separates the pile of building blocks. That is, "A-B" and "C-D" are two piles of building blocks. Then a PN model (Figure 7) can be automatically

generated by TS_2, where places are represented by circles, transitions are represented by rectangles and the red place (p2, p3, p5, p8) means the place containing one token. It's obvious that the operation USPD (E, F) can be performed in state s0 to reach the state s10. That is, this supplement is in line with reality, so as others. Consequently, the information is increased reasonably.



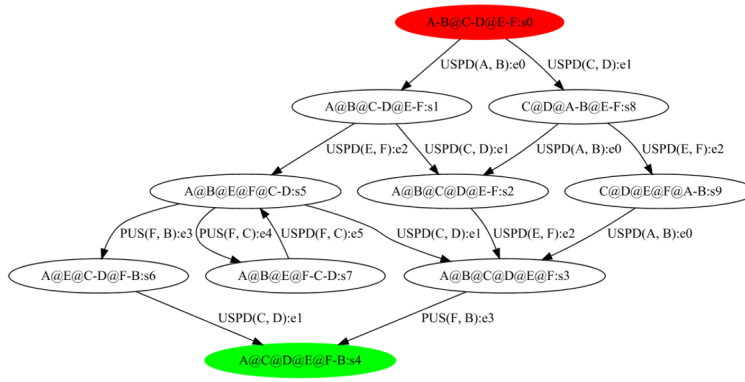
a) b)

Figure 5. a) Initial state and b) target state of the building blocks

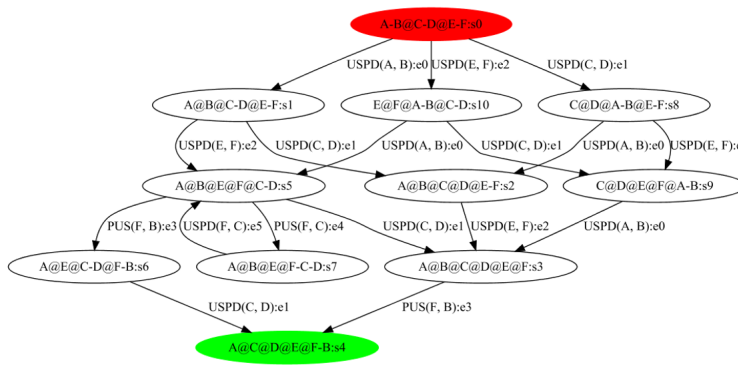
Then, in order to demonstrate the effectiveness of the algorithm better, we design three more complex examples and simulate them. The initial state and target state of the examples are shown in Table 2. In the simulation, we specify that the PN model is going to be generated when the program finds a path from the initial state to the target. For each example, we compare the results of the exhaustive generation and supplementary generation by program. The exhaustive generation means the generation of a PN model according to the known information directly and the supplementary generation means the generation of a PN model via using Algorithm 1 which includes the information added steps. The results are shown in Table 3. For exhaustive generation, the number of attempts to generate a PN model as well as the failure times is computed during the period of generating a complete TS. For supplementary generation, the number of attempts to generate a PN model by Algorithm 1, the states and arcs added to the TS during the whole process are calculated. The total number of states and arcs of TS are also listed in the table. Definitely, we can avoid the failures of model generation thanks to Algorithm 1, so there are no failure times in supplementary generation. It can be seen that the attempt times of supplementary generation are less than exhaustive generation for all examples listed in Table 3. Even more, supplementary generation can add some states and arcs to TS which lead to the reduction in attempt times. In other words, by using the proposed algorithm, we can obtain a complete PN model without traversing completely. Consequently, it is available to learn a PN model based on the method presented in this paper.

	Initial State	**Target State**
1	A-B@C-D@E-F	A@B-C@F@D-E
2	A-B-C@D-E	E-C-B@A-D
3	A@B@G-C@D@F-E	A-B@D-C@E@F@G

Table 2. Examples of simulations



a)



b)

Figure 6. a) The initialized transition system TS_3 which is produced by robot's attempts, b) the initialized transition system TS_4 after adding information to TS_3

5 CONCLUSIONS

With the development of intelligent technology, more and more researchers begin to join in the field of robotics and devote to the modeling and learning of the robot system. PN is an abstract formal modeling method, which can represent the sequence and concurrent events as well as the restrictions of various conditions. In view of the flexibility and effectiveness, PN can be applied to the robot field. Motivated by the scene of building block world, this paper introduced two theorems and a PN model generation algorithm based on region theory, which achieves a PN model generated automatically according to a transition system, as well as makes model more complete to some extent. Besides, the effectiveness of the method is demonstrated by a program which simulates the robot scene and applies the algorithm.

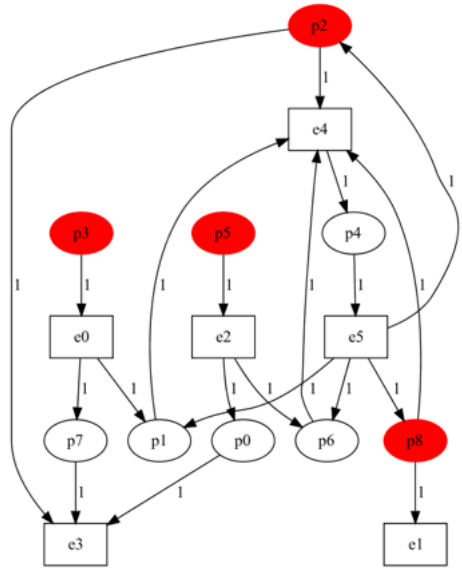


Figure 7. The corresponding PN of TS_4

	Exhaustive Generation		Supplementary Generation				
	Attempt Times	Failure Times	Attempt Times	Number of Added States	Number of Added Arcs	Total States	Total Arcs
1	90	33	75	6	24	40	90
2	243	106	183	28	95	98	249
3	175	85	128	15	65	62	171

Table 3. Results of simulations

In future work, we intend to improve the performance of the algorithm as well as studying the model analysis methods. Furthermore, we will focus on the extension of the method to multi-robot systems and other automated manufacturing systems.

Acknowledgement

This work is partially supported by the National Key Research and Development Program of China under Grant No. 2018YFB2100801 and by the National Natural Science Foundation of China under Grant No. 61672381, and in part by the Fundamental Research Funds for the Central Universities under Grant No. 22120180508.

REFERENCES

[1] Desai, J. P.: A Graph Theoretic Approach for Modeling Mobile Robot Team Formations. Journal of Robotic Systems, Vol. 19, 2002, No. 11, pp. 511–525, doi: 10.1002/rob.10057.

[2] Wieber, P.-B.—Tedrake, R.—Kuindersma, S.: Modeling and Control of Legged Robots. In: Siciliano, B., Khatib, O. (Eds.): Springer Handbook of Robotics. Springer Handbooks, Springer, Cham, 2016, pp. 1203–1234, doi: 10.1007/978-3-319-32552-1_48.

[3] Sim, S. K.—Ong, K. W.—Seet, G.: A Foundation for Robot Learning. 2003 4th International Conference on Control and Automation Proceedings, IEEE, 2003, pp. 649–653, doi: 10.1109/ICCA.2003.1595102.

[4] Norris, D. J.: Behavior-Based Robotics. Chapter 11. In: Norris, D. J.: Beginning Artificial Intelligence with the Raspberry Pi. Apress, Berkeley, CA, 2017, pp. 313–345, doi: 10.1007/978-1-4842-2743-5_11.

[5] Brooks, R. A.—Mataric, M. J.: Real Robots, Real Learning Problems. In: Connell, J. H., Mahadevan, S. (Eds.): Robot Learning. Springer, Boston, MA, The Springer International Series in Engineering and Computer Science (Knowledge Representation, Learning and Expert Systems), Vol. 233, 1993, pp. 193–213, doi: 10.1007/978-1-4615-3184-5_8.

[6] Demiris, J.—Birk, A.: Interdisciplinary Approaches to Robot Learning: Introduction. World Scientific Series in Robotics and Intelligent Systems, Vol. 24, 2000, pp. 1–7, doi: 10.1142/9789812792747_0001.

[7] Murata, T.: Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE, Vol. 77, 1989, No. 4, pp. 541–580, doi: 10.1109/5.24143.

[8] Heiner, M.—Herajy, M.—Liu, F.—Rohr, C.—Schwarick, M.: Snoopy – A Unifying Petri Net Tool. In: Haddad, S., Pomello, L. (Eds.): Application and Theory of Petri Nets (PETRI NETS 2012). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 7347, 2012, pp. 398–407, doi: 10.1007/978-3-642-31131-4_22.

[9] Lima, P.—Gracio, H.—Veiga, V.—Karlsson, A.: Petri Nets for Modeling and Coordination of Robotic Tasks. 1998 IEEE International Conference on Systems, Man, and Cybernetics (SMC'98), San Diego, CA, USA, 1998, Vol. 1, pp. 190–195, doi: 10.1109/ICSMC.1998.725407.

[10] Ziparo, V. A.—Iocchi, L.: Petri Net Plans. Proceedings of Fourth International Workshop on Modelling of Objects, Components, and Agents (MOCA), June 2006, pp. 267–290.

[11] Chao, C.—Thomaz, A. L.: Timing in Multimodal Turn-Taking Interactions: Control and Analysis Using Timed Petri Nets. Journal of Human-Robot Interaction, Vol. 1, 2012, No. 1, pp. 4–25, doi: 10.5898/JHRI.1.1.Chao.

[12] Chang, G.—Kulić, D.: Robot Task Learning from Demonstration Using Petri Nets. 2013 IEEE International Workshop on Robot and Human Communication (ROMAN), Gyeongju, South Korea, 2013, pp. 31–36, doi: 10.1109/ROMAN.2013.6628527.

[13] van der Aalst, W.—Weijters, T.—Maruster, L.: Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering, Vol. 16, 2004, No. 9, pp. 1128–1142, doi: 10.1109/TKDE.2004.47.

[14] Roldán, J. J.—del Cerro, J.—Barrientos, A.: Using Process Mining to Model Multi-UAV Missions Through the Experience. IEEE Intelligent Systems, Vol. 32, 2017, No. 4, pp. 40–47, doi: 10.1109/MIS.2017.3121547.

[15] Badouel, E.—Bernardinello, L.—Darondeau, P.: Petri Net Synthesis. Springer, Berlin, Heidelberg, Texts in Theoretical Computer Science, 2015, doi: 10.1007/978-3-662-47967-4.

[16] Badouel, E.—Darondeau, P.: Theory of Regions. In: Reisig, W., Rozenberg, G. (Eds.): Lectures on Petri Nets I: Basic Models (ACPN 1996). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 1491, 1996, pp. 529–586, doi: 10.1007/3-540-65306-6_22.

[17] Cortadella, J.—Kishinevsky, M.—Kondratyev, A.—Lavagno, L.—Yakovlev, A.: Petrify: A Tool for Manipulating Concurrent Specifications and Synthesis of Asynchronous Controllers. IEICE Transactions on Information and Systems, Vol. E80-D, 1997, No. 3, pp. 315–325.

[18] Carmona, J.—Cortadella, J.—Kishinevsky, M.: Genet: A Tool for the Synthesis and Mining of Petri Nets. 2009 Ninth International Conference on Application of Concurrency to System Design, Augsburg, Germany, July 2009, pp. 181–185, doi: 10.1109/ACSD.2009.6.

[19] Badouel, E.—Caillaud, B.—Darondeau, P.: Distributing Finite Automata Through Petri Net Synthesis. Formal Aspects of Computing, Vol. 13, 2002, No. 6, pp. 447–470, doi: 10.1007/s001650200022.

[20] Best, E.—Schlachter, U.: Analysis of Petri Nets and Transition Systems. Proceedings of the 8th Interaction and Concurrency Experience Workshop (ICE 2015). Electronic Proceedings in Theoretical Computer Science (EPTCS), Vol. 189, 2015, pp. 53–67, doi: 10.4204/EPTCS.189.6.

[21] Wu, Z. H.: Introduction to Petri Nets. China Machine Press, Beijing, 2006 (in Chinese).

[22] Best, E.—Devillers, R.—Schlachter, U.: Bounded Choice-Free Petri Net Synthesis: Algorithmic Issues. Acta Informatica, Vol. 55, 2018, No. 7, pp. 575–611, doi: 10.1007/s00236-017-0310-9.

[23] Schlachter, U.: Petri Net Synthesis for Restricted Classes of Nets. In: Kordon, F., Moldt, D. (Eds.): Application and Theory of Petri Nets and Concurrency (PETRI NETS 2016). Springer, Cham, Lecture Notes in Computer Science, Vol. 9698, 2016, pp. 79–97, doi: 10.1007/978-3-319-39086-4_6.

[24] Wolf, K.: Petri Net Synthesis with Union/Find. In: Khomenko, V., Roux, O. (Eds.): Applications and Theory of Petri Nets and Concurrency (PETRI NETS 2018). Springer, Cham, Lecture Notes in Computer Science, Vol. 10877, 2018, pp. 60–81, doi: 10.1007/978-3-319-91268-4_4.

[25] Tredup, R.: Hardness Results for the Synthesis of b-Bounded Petri Nets. In: Donatelli, S., Haar, S. (Eds.): Applications and Theory of Petri Nets and Concurrency (PETRI NETS 2019). Springer, Cham, Lecture Notes in Computer Science, Vol. 11522, 2019, pp. 127–147, doi: 10.1007/978-3-030-21571-2_9.

Jiao Li received her B.Sc. degree in computing science and technology from the Jiangsu University, Zhenjiang, China, in 2017. She is currently pursuing her M.Sc. degree with the Department of Computer Science and Technology, Tongji University, Shanghai, China. Her current research interests include Petri nets and formal engineering.

Ru Yang received her B.Sc. degree from the Shandong University of Science and Technology, Qingdao, China, in 2013. She is currently pursuing her Ph.D. degree with the Department of Computer Science and Technology, Tongji University, Shanghai, China. Her current research interests include Petri nets and formal engineering.

Zhijun Ding received his M.Sc. degree from the Shandong University of Science and Technology, Tai'an, China, in 2001, and the Ph.D. degree from Tongji University, Shanghai, China, in 2007. He serves currently as Professor with the Department of Computer Science and Technology, Tongji University. He has published over 100 papers in domestic and international academic journals and conference proceedings. His research interests are in formal engineering, Petri nets, services computing, and mobile internet.

Meiqin Pan received her Ph.D. degree from Shandong University of Science and Technology, Qingdao, China, in 2008. Now she is Associate Professor of the School of Business and Management, Shanghai International Studies University. Her research interests are in information systems, data mining and technology optimization methods. She has published more than 20 papers in domestic and international academic journals and conference proceedings.