Formal Modeling of Hybrid System Based on Semi-continuous Colored Petri Net: A Case Study of Adaptive Cruise Control System

Formal Modeling of Hybrid System Based on Semi-continuous Colored Petri Net: A Case Study of Adaptive Cruise Control System

WANGYANG YU, QI GUO*, and YUMENG CHENG, The School of Computer Science, Shaanxi Normal University, China
LU LIU, Department of Computer Science, University of Exeter, UK
FEI HAO, The School of Computer Science, Shaanxi Normal University, China
XIAOJUN ZHAI, School of Computer Science and Electronic Engineering, University of Essex, UK
MINSI CHEN, School of Computing and Engineering, University of Huddersfield, UK

Many Next-Generation consumer electronic devices would be distributed hybrid electronic systems, such as UAVs (Unmanned Aerial Vehicles) and smart electronic cars. The safety and risk control are the key issues for the sustainability of such consumer electronic systems. The modeling of hybrid electronic systems is difficult to be abstracted by traditional Petri Nets. This also makes the reachable marking graph unable to be applied to Petri nets of the hybrid electronic systems. This paper proposes a novel Petri Net to model and analyze the hybrid electronic systems. We name it a Semi-continuous Colored Petri Net (SCPN) that inherits the excellent modeling capabilities and analysis methods of Petri Nets, and can formally depict hybrid quantities. In addition, we propose the construction algorithm for an SCPN reachable marking graph and prove its finiteness. Finally, we model and analyze an Adaptive Cruise Control (ACC) system of smart electronic cars as an example to prove the validity of SCPN. We use the proposed SCPN to model and analyze the running process of an ACC system under the continuous deceleration scenario of the front vehicle. The application study shows that the ACC system has logic flaws under the constant headway strategy when the front vehicle continues to decelerate. Based on this analysis, improvements to the SCPN of the ACC system are made, effectively enhancing its safety and logical correctness.

CCS Concepts: • **Theory of computation → Formal languages and automata theory**; • **Computer systems organization → Embedded systems**.

Additional Key Words and Phrases: Petri nets, Dynamic network modeling, Modeling and simulation, Hybrid systems, Formal model

---

This work was in part supported by the Open Research Fund of Anhui Province Engineering Laboratory for Big Data Analysis and Early Warning Technology of Coal Mine Safety, China, under Grant No. CSBD2022-ZD05, in part by the Open Research Fund of Key Laboratory of Embedded System and Service Computing (Tongji University), Ministry of Education, China under Grant ESSCKF2023-02, in part by the Fundamental Research Funds for the Central Universities under Grant No. GK202205039, and in part by the National Natural Science Foundation of China, under Grant No. 62477029. This work was also supported by the China Scholarship Council.
Authors' addresses: Wangyang Yu, ywy191@snnu.edu.cn; Qi Guo, gqojbk@163.com; Yumeng Cheng, chengym@snnu.edu.cn, The School of Computer Science, Shaanxi Normal University, Xi'an, Shaanxi, China, 710119; Lu Liu, Department of Computer Science, University of Exeter, Exeter, UK, l.liu3@exeter.ac.uk; Fei Hao, The School of Computer Science, Shaanxi Normal University, Xi'an, China, fhao@snnu.edu.cn; Xiaojun Zhai, School of Computer Science and Electronic Engineering, University of Essex, Colchester, UK, xzhai@essex.ac.uk; Minsi Chen, School of Computing and Engineering, University of Huddersfield, Huddersfield, UK, m.chen@hud.ac.uk.

---

1 INTRODUCTION

In our daily life, precise control of hybrid information systems plays a key role in many industries due to their safety and robustness. To ensure the safety and logical correctness of such systems, it is necessary to introduce standardized formal methods into the system development process. Formal methods can improve the accuracy and robustness of software and hardware design using appropriate mathematical analysis techniques, ensuring a high degree of integration and validation in system development.

Some researchers use architecture analysis and design language to carry out half-formal modeling of embedded systems to improve their safety and operational accuracy[20, 27, 29, 30, 35]. Hybrid automata is a classical formal method of model checking, which is widely used in hybrid systems for modeling and verification [10, 15, 17]. However, there are problems such as state explosion in hybrid automata [11, 19, 34], which makes the analysis of hybrid automata models intractable. The differential dynamic logic proposed by Platzer combines synthetic verification techniques and modularization of verification problems, which provides a strong theoretical basis for the modeling and verification of hybrid systems [25].

A Petri net is a graphical formal model, which can describe both a structure of the system and its inherent concurrent and conflicting processes within[36]. For hybrid systems, Petri net can describe them hierarchically, which is combined with object-oriented ideas in software development. The above features make it easier to understand by developers and testers. Meanwhile, a Petri net is also used in various fields such as medical treatment, e-commerce, transportation, military and so on [12, 21, 26, 31]. T. Chakraborty applied Petri nets to model and analyze cloud platform fleet ACC. Experimental results showed that priority Petri nets could effectively reduce the accumulation of cloud platform data [2]. Chandramohan used an original Petri net to model and analyze the ACC system, and established a fault-tolerant mechanism for accelerator and braking [3]. Carlos Gmez-Huelamo used hierarchical interpreted binary Petri nets to describe the decision-making framework of autonomous driving in the ACC scenario [13].

Despite the aforementioned successes in its application, there have been a number of well reported limitations which restrict its extension onto more complex dynamical systems. First of all, the original Petri net has two characteristics of high abstraction and discrete nature of the net model, which makes it extremely difficult to use the original Petri net to model for hybrid systems [1, 31]. Several extensions devised to address this limitation have been proposed by researchers in the past decade [1, 7, 28, 32, 33]. These high-level Petri nets can more accurately depict the basic principles and control procedures, realize the formal construction, and conduct the risk analysis to improve the validity and safety of hybrid systems.

A Colored Petri Net (CPN) is a common high-level Petri net[6, 8, 16, 22, 23], which has powerful modeling ability due to the combination with ML language. Meanwhile, CPN is a discrete event modeling language, which is used to describe the discrete events in dynamical systems, and plays an important role in concurrency, synchronization, and communication systems. In addition, the CPN description of a system also closely resembles its implementation. This makes CPN suitable for modeling of concurrent engineering systems. However, CPN's inability to adequately representing continuous changes has remained a challenge when modeling hybrid systems. Due to this, it is difficult for us to extract process description and data for further analysis and verification. This

requires us to modify the CPN, while retaining its strong modeling capabilities, to make it more complete the depiction of continuous changes.

A stochastic Petri net (SPN) is a special high-level net that gives each transition a firing rate that can be thought of as random variable X drawn from a distribution function [4, 7, 14, 18, 28]. The distribution function can be obtained empirically from actual measurement results, or by generating a prediction model according to predefined rules. The latter approach enhances the SPN's simulation capabilities to better fit and reflect the running mechanism of the actual system. Either discrete or continuous transition firing rate can be used in SPN. Although this extension allows a more robust description of transition control, there is still no means of describing the change of token inside a place. Hence, SPN is essentially a Petri net for modeling discrete systems.

Hybrid Petri Net (HPN) is a high-level Petri net emphasising on depicting continuous changes [5, 9, 24, 28]. Basile combines HPN with CPN and proposed Colored Modified Hybrid Petri Nets to model complex material handling systems [1]. The advantage of this is that the accumulation of continuous quantities can be calculated by definite integral. However, in the HPN, discrete and continuous places separately participate in the firing process of transitions. In other words, the sets for discrete and continuous places are disjoint. First, in a HPN, discrete places and transitions are independent of continuous ones. As a result of this, the token in the continuous place cannot participate in the firing process of discrete transitions. The token in the discrete place conversely cannot participate in the firing process of continuous transitions. This is inconsistent with the operation of real-world systems. In addition, HPN cannot be analyzed by the reachable marking graph of Petri nets. Since the change of the continuous quantity is infinite, the reachable marking graph of the HPN and the continuous Petri net is also an infinite set, which cannot be analyzed directly. This means that analysis methods such as reachable marker graph cannot be applied on the HPN.

To further model and analyze the running process of a system, this paper proposes Semi-continuous Colored Petri Net (SCPN). SCPN is based on CPN's modeling capabilities, making it easy to implement the system. SCPN also introduced the concept of hybrid Petri nets to depcit continuous quantities with additional improvements to make SCPN more suitable for the modeling and analyzing a hybrid system. The main idea of SCPN is to replace the weight of the arc with the average rate of continuous quantity change. The advantage of this is that the continuous quantity can be evaluated so that the reachable marking graph can be used for the SCPN of a hybrid systems. In summary, the proposed model can combine data analysis and process depiction to improve the interpretability and traceability of a formal models.

In this paper, the design of the distance algorithm of an ACC system is taken as an example, and it is verified in its conceptual model stage using the proposed SCPN. The main contributions of this paper are summarized as follows:

(1) A novel extension to Petri net, SCPN, is proposed to model the hybrid systems.
(2) A reliable analysis method of the SCPN model is outlined.
(3) We provide a novel modeling scheme for hybrid systems.

The rest of this paper is organised as follows: The section II introduces the basic concepts of SCPN. The section III presents the analysis method of SCPN and proves its correctness. The section IV uses the modeling and analysis of the ACC system as a use case to verify the validity of SCPN. Section V provides a related discussion. Section VI concludes this paper.

2 BASIC CONCEPTS

SCPN can combine the intuitiveness of graph representations with the logic of mathematics, and carry out modeling and analysis of the hybrid systems that possesses discreteness and continuity

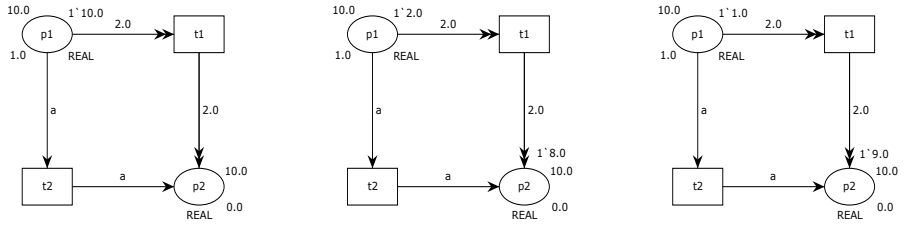

Fig. 1. The running example of the continuous transition in SCPN.

at the same time. The definition of SCPN is given as follows:
Definition 1 : An SCPN is a ten-tuple, SCPN = $(P, T, A, \Sigma, V, C, G, E, I, K)$ where:

(1) P is a finite set of places, $P_C \cup P_D = P$, $P_C \cap P_D = P_H$, where P_C represents continuous place, P_D represents discrete place, P_H represents hybrid place.
(2) T is a finite set of transitions, $T_C \cap T_D = T_H$, $T_C \cup T_D = T$, where T_C represents continuous transition, T_D represents discrete transition, T_H represents hybrid transition, which is divided into T_H^i and T_H^o, T_H^i represents hybrid transition where the input is continuous and the output is discrete, T_H^o represents hybrid transition where the input is discrete and the output is continuous.
(3) $A \subseteq \{P \times T\} \cup \{T \times P\}$ is a set of directed arcs, $A_C \cap A_D = \emptyset$, $A_C \cup A_D = A$, where A_C represents continuous arc, A_D represents discrete arc.
(4) Σ is a finite set of non-empty continuous colour sets.
(5) V is a finite set of typed variables such that $Type[v] \in \Sigma$ for $\forall v \in V$.
(6) $C : P \rightarrow \Sigma$ is a continuous colour set function that assigns a continuous colour set to each place.
(7) $G : T \rightarrow EXPR_V$ is a guard function that assigns a guard to each transition t such that $Type[G(t)] = Bool$.
(8) $E : A \rightarrow EXPR_V$ is an arc expression function that assigns an arc expression to each arc a, where $||E(p,t)||$ represents the weight on the arc.
(9) $I : P \rightarrow Q^+$ is an marking function that record the token of each place. I_0 assigns an initialisation marking to each place p such that $Type[I(p)] = C(p)_{MS}$, where Q^+ represents positive Rational number.
(10) $K : P \rightarrow Q^+$ is a capacity function, which defines the upper and lower bounds of each continuous places such that $Type[K(P_C)] = C(p)_{MS}$, where K_{low} represents lower bounds of continuous places, K_{high} represents upper bounds of continuous places.

Definition 2 : The rules of firing continuous transitions:

(1) For $\forall t \in T_C$, the condition of $L[t >$ is:

1. $\forall p \in {}^\bullet t : L(p) \geq K(p)_{low} \wedge L(p) \geq ||E(p,t)||$

2. $\forall p \in t^\bullet - {}^\bullet t : K(p)_{low} \leq L(P) + ||E(t,p)|| \leq K(p)_{high}$

3. $\forall p \in t^\bullet \cap {}^\bullet t : K(p)_{low} \leq L(P) + ||E(t,p)|| - ||E(p,t)|| \leq K(p)_{high}$

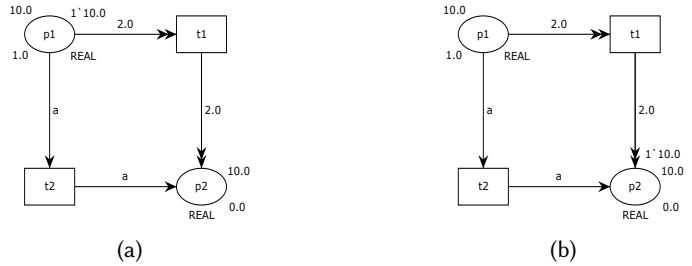

Fig. 2. The running example of the discrete transition in SCPN.

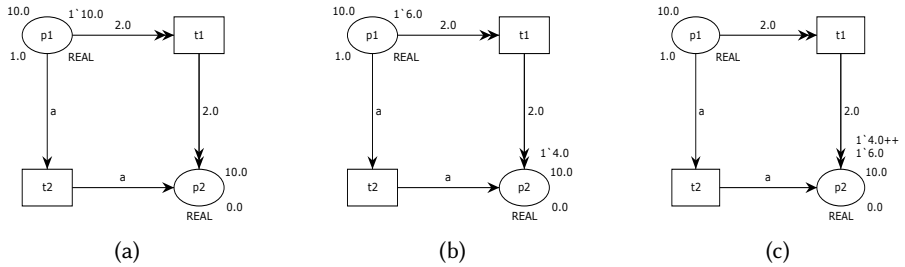

Fig. 3. The running example of the discrete&continuous transition in SCPN.

(2) If $L[t > L'$, then to $\forall p \in P$:

$$L'(p) = \begin{cases} L(p) - ||E(p,t)||, p \in {}^\bullet t - t^\bullet \\ K(p)_{low}, \text{if } L(p) - ||E(p,t)|| \leq K(p)_{low} \\ L(p) + ||E(t,p)||, p \in t^\bullet - {}^\bullet t \\ K(p)_{high}, \text{if } L(p) + ||E(t,p)|| \geq K(p)_{high} \\ L(p) + ||E(t,p)|| - ||E(p,t)||, p \in {}^\bullet t \cap t^\bullet \\ L(p), others \end{cases}$$

Definition 3 : The rules of firing hybrid transitions:

(1) For $\forall t \in T_H^i$, the condition of $L[t >$ is:

$$\forall p \in {}^\bullet t : L(p) \geq K(p)_{low} \wedge L(p) \geq ||E(p,t)||$$

If $L[t > L'$, then to $\forall p \in {}^\bullet t$:

$$L'(p) = \begin{cases} L(p) - ||E(p,t)||, p \in {}^\bullet t - t^\bullet \\ K(p)_{low}, \text{if } L(p) - ||E(p,t)|| \leq K(p)_{low} \end{cases}$$

If $L[t > L'$, then to $\forall p \in t^\bullet$:

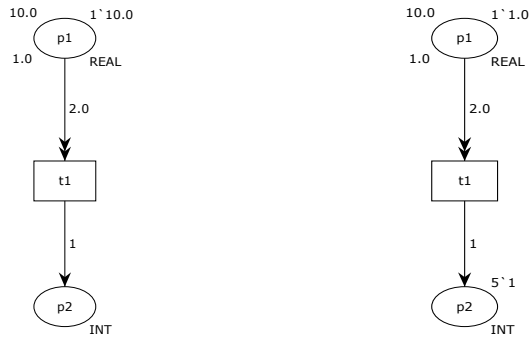

Fig. 4. The running example of the hybrid (continuous input and discrete output) transition in SCPN.

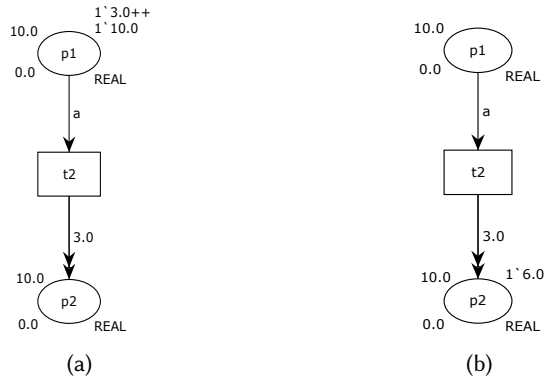

Fig. 5. The running example of the hybrid (discrete input and continuous output) transition in SCPN.

$$L'(p) = L(p) + +||E(p,t)||, p \in t^{\bullet} - {}^{\bullet}t$$

where "+ +" represents the operative symbol of multiset in the colored Petri nets [16], also indicating here the flow of tokens.

(2) For $\forall t \in T_H^o$, the condition of $L[t >$ is:

$$\forall p \in {}^{\bullet}t : L(p) \geq ||E(p,t)||$$

If $L[t > L'$, then to $\forall p \in {}^{\bullet}t$:

$$L'(p) = L(p) - -||E(p,t)||, p \in {}^{\bullet}t - t^{\bullet}$$

If $L[t > L'$, then to $\forall p \in t^{\bullet}$:

$$L'(p) = \begin{cases} L(p) + ||E(t,p)||, p \in t^{\bullet} - {}^{\bullet}t \\ K(p)_{high}, \text{ if } L(p) + ||E(t,p)|| \geq K(p)_{high} \end{cases}$$

where "− −" represents the flow of token.

Definition 4 : Let H be an SCPN, the reachable marking graph of SCPN (SRMG) is a three-tuple, $SRMG = \{R(I_0),$
$L, Q\}$ where:

(1) $R(I_0)$ is the set of all reachable marking of SCPN.
(2) $L = \{(I_i, I_j)|I_i, I_j \in R(I_0), \exists t_k \in T$ make $I_i[t_k > I_j\}$.
(3) $Q = \{t_k|t_k \in T, L \rightarrow T\}$.

$R(I_0)$ is the reachable marking set, L is the arc set, and Q is the corresponding transition set on the L.

SCPN divides a continuous change into multiple fragmented changes. SCPN's simulation of continuous changes is established when the granularity of the division of this fragment is small enough. Fig. 1 is a simple schematic diagram of SCPN running example. Hybrid places can not only participate in the firing of continuous transitions, but also participate in the firing of discrete ones. Fig. 1a shows the initial marking of the example. p_1 and p_2 are hybrid places which means they are both continuous and discrete places. At this time, both t_1 and t_2 are enabled, where t_1 is a continuous transition and t_2 is a discrete one. The different firing order of t_1 and t_2 will lead to different results. In addition, due to the limitation of the capacity function $K(P)$, the firing of t_1 and t_2 will be subject to certain constraints. In Fig. 1b, the value of the token in p_1 is greater than $K(p_1)_{low}$ after t_1 firing four times. According to the Definition 2, t_1 is still in the enabled state that means t_1 can continue to fire. As some of the tokens in p_1 are consumed, the value of the remaining tokens is determined by the lower bound of the capacity function. The result is shown in Fig. 1c. In order to better illustrate the characteristics of SCPN, the example adopts a segmentation granularity.

Fig. 2 is a schematic diagram of the firing of the discrete transition t_2 following the firing rule of CPN transitions. It is worth noting that the firing of discrete transitions is not restricted by the capacity function. This is due to the setting of the capacity function that prevents continuous quantity overflow. In the modeling of an ACC system, the firing of discrete transitions has no influence on the value of the continuous quantity. Thus, the firing of continuous transitions is governed by the capacity function.

The simultaneous firing of continuous transition t_1 and discrete transition t_2 are shown in Fig. 3. Fig. 3b shows the results after t_1 firing twice when t_2 is in the enabled state. After t_2 firing, two tokens are generated at p_2. One is a brand new token generated at p_2 by the continuous transition t_1. It is a brand new token generated by the continuous transition t_1. The other is a token transformed from its initial marking by the the firing of the discrete transitions t_2. We used different arc shapes to differentiate the two types of hybrid transition. An arc with double arrows indicates that the token continuously participates in the firing of the transition, and the arc of the single arrow arc indicates that the token discretely participates in the firing of the transition, see Fig. 4 and Fig. 5. In Fig. 4a, input arc of transition t_2 is continuous, while the output arc is discrete. Under this marking, the input arc of t_1 follows the rule of the continuous transition, and the output arc follows the rule of discrete one. The running result is shown in Fig. 4b. The actual meaning represented by the discrete place p_2 is to count the number of changes in the continuous quantity. Similarly, Fig. 5 shows an example where the input arc of the transition is discrete and the output arc is continuous. In Fig. 5a, there are two tokens in p_1, and the firing of t_1 will consume one of them. t_1 becomes disable status after consuming all the tokens and cannot continue to fire. Since t_1 fires twice and

consumes the two tokens in p_1, a new token is generated in p_2, and it is incremented twice. Such an example takes one or more color sets as input, and the continuous change of token is the output.

3 VERIFICATION AND ANALYSIS

The classic analysis method of Petri nets is the reachable marking graph. The reachable marking graph can express all the running states of Petri nets and judge various dynamic properties of Petri nets. These dynamic properties also reflect the characteristics of the actual system. However, for hybrid Petri nets and continuous Petri nets, there is no way to analyze them with typical Petri net analysis methods such as the reachable marking graph. This is because the reachable marking graph of continuous Petri nets and hybrid Petri nets are infinite, and cannot be exhaustively listed, which makes the analysis of such Petri nets more difficult. The SCPN proposed in this paper can use the reachable marking graph to analyze the changes of the continuous quantity and apply it to the actual system.

When the place P_j is unbounded, its number of mark may increase indefinitely during the running of the Petri net. At this time, the j vector in the mark vector is changed to ω to cover all such marks. Therefore, we can use a finite tree to represent the infinite running of a Petri net. The construction algorithm of the SRMG is given in Algorithm 1.

The time complexity of this Algorithm 1 can be determined by analyzing the number of operations in each step. The time complexity of this algorithm can be determined by analyzing the number of operations in each step. Initially, the operation of selecting the root node I_0 and marking it as "new" has a constant time complexity of $O(1)$. Next, the while loop iterates over all "new" nodes in the SRMG. In the worst case, this loop will execute n iterations, where n is the number of nodes in the SRMG. During each iteration, selecting a "new" node and setting it as I also has a time complexity of $O(1)$. Subsequently, the algorithm checks whether there is a path from I_0 to I and marks the node as "old." In the worst case, this operation requires traversing the paths of all previously visited nodes, leading to a time complexity of $O(n-1)$, which can be simplified to $O(n)$. If none of the transitions t can be fired at node I, the algorithm marks the node as a "terminal node," with a time complexity of $O(1)$. For each node I, the algorithm computes the number of new nodes I' and creates edges. The time complexity of this step is related to the number of nodes $|I'|$ in I', and in the worst case, it is $O(|I'|)$. Afterward, the algorithm checks for satisfying paths and updates the nodes. The time complexity for path checking depends on the number of paths already traversed. In the worst case, all previous paths need to be traversed, resulting in a time complexity of $O(n)$. Finally, the operations of removing the old "new" label and marking new nodes have a time complexity of $O(1)$. Therefore, the overall worst-case time complexity of the algorithm is $O(n^2)$, as the while loop executes n times, with the worst-case complexity of operations in each iteration being $O(n)$.

For $\forall p \in P$, If $K(p) = \emptyset$, Then $R(I)$ may be an infinite set. Therefore, the SRMG will also be infinite. In order to use a finite form to represent the hybrid systems in an infinite running state, it is necessary to introduce an unbounded quantity ω.
ω has such properties:

(1) For arbitrary positive rational numbers $s : \omega > s$, $\omega \pm s = \omega$.
(2) $\omega \geq \omega$.

An algorithm should have finiteness, accuracy, input, output and feasibility. Except for finiteness, the other four properties are obvious. This section gives a proof of finiteness. Before that, first introduce related concepts.
Definition 5: An infinite directed tree is a two-tuple, $\Omega = (H, L)$ where H is an infinite set of nodes, L is an infinite set of arcs, $H \cap L = \emptyset$.
Regulation:

Algorithm 1: Construction Algorithm of SRMG.

Input: $SCPN = (P, T, A, \Sigma, V, C, G, E, I, K)$, initial marking I_0
Output: SRMG
1.Take I_0 as the root node of SRMG and label it as "new";
2.**while** ∃ *"new" nodes* **do**
  Choose an arbitrary "new" node and set it to I;
  if ∃ *node = I from* I_0 *to I* **then**
    Label it as "old";
    Return to step2;
  if $\forall t \in T : \neg I[t >$ **then**
    Label I as "terminated node";
    return to step2;
  for $I[t >$ **do**
    Calculate I' in $I]t > I'$;
    Create a directed edge from I to I';
    Mark the edge labeled with t;
    if $\exists I''$ *make* $I'' < I'$ *from* I_0 *to* I' **then**
      Find j make $I''(p_j) < I'(p_j)$;
      if $K(p_j) = \emptyset$ **then**
        Change the j component of I' to ω;
    Label I' as "new"
    Remove label "new" of I;
    Return step2;

(1) $p \subseteq \Omega$ is an infinite directed path.
(2) $\Omega' \subseteq \Omega$ is an infinite directed subtree.
(3) $Suc(v) \in V$ is the direct successor nodes of v.
(4) $Pre(v) \in V$ is the direct precursor nodes of v.
(5) $Root(\Omega)$ is the root node of Ω.
(6) M is an arbitrary non-negative finite rational number.
(7) s is an infinite sequence.
(8) s^+ is an infinite increasing sequence, which meet for $\forall i, j \in N^+(i > j), s_i^+ > s_j^+$.
(9) s^- is an infinite decreasing sequence, which meet for $\forall i, j \in N^+(i > j), s_i^- < s_j^-$.

For this, we first prove the following three propositions.

Proposition 1: In Ω, for $\forall v \in V$, **if** $Suc(v)$ is finite, **Then** ∃ an infinite directed path p starting from the $Root(\Omega)$.

Proof: In Ω, because $Suc(Root(\Omega))$ is finite, so at least one of the $Suc(Root(\Omega))$ is $Root(\Omega')$ of infinite Ω'. If $\forall\Omega'$ rooted at $Suc(Root(\Omega))$ are finite, then Ω is finite, which contradicts the premise. In the same way, $Suc(Root(\Omega'))$ is finite, at least one of $Suc(Root(\Omega'))$ is $Root(\Omega'')$ of infinite Ω''. According to this logic, we can get an infinite directed path p starting from $Root(\Omega)$.

Proposition 1 is proved. □

Proposition 2: For $\forall s$ composed of M, $\exists s' \subseteq \complement_s s^-$, where $\complement_s s^-$ represents the complement set of s^- in s.

Proof: Proposition 2 is proved in two cases:

(1) When $\exists m (m \in M)$ that appears infinitely in s': s' is the subsequence $\{m, m, m...\}$.

(2) When $\forall m(m \in M)$ that appears finitely in s': let $s' \subseteq s^-$, because s' is infinite and decreasing, $\exists$ infinite $s'_i \in s'$, making $s'_i < m$. s'_i is decreasing and $s'_i \geq 0$, s'_i must be a finite rational numbers, which contradicts the premise of Proposition 2, so $\exists s' \subseteq \complement_s s^-$.

Proposition 2 is proved. □

Proposition 3: x is an n-dimensional vector composed of M or ω. For $\forall s$ composed of x , $\exists s' \subseteq \complement_s s^-$.
Proof: We can use mathematical induction to prove it.

(1) When n=1 : if x composed of infinite ω, then $s' \subseteq \complement_s s^-(\omega \geq \omega)$. if x composed of m and finite ω, we remove the finite x composed of ω. At this time, The vector x is equivalent to M. According to Proposition 2, $\exists s' \subseteq \complement_s s^-$.

(2) Assuming that Proposition 3 is true for n-dimensional vectors x, it is now proved that Proposition 3 is also true for $(n+1)$-dimensional vectors x'. For the first component of a $(n+1)$-dimensional vector x': If $\exists$ infinite x' that meet $x'_1 = \omega$, then $s' \subseteq \complement_s s^-$ (s' composed of x'_1). If $x'_1 \neq \omega$, then according to Proposition 2, $\exists$ subsequence $s' \subseteq \complement_s s^-$ (s' composed of x'_1). Now, we ignore the first component x'_1, and consider the remaining n component in x'_1 as x. According to the above induction hypothesis, s composed of x $\exists$ subsequence $s' \subseteq \complement_s s^-$. When the s' is determined, and the first component x'_1 is added, we get a non-decreasing $(n+1)$-dimensional vector sequence $s'' \subseteq \complement_s s^-$.

Proposition 3 is proved. □

Now we can prove the finiteness of the SRMG construction algorithm. Assume that SRMG is an infinite reachable marking graph Ω_1. According to Proposition 1, $\exists$ an infinite directed path p starting from I_0 (Each node I has at most t direct successors, and t is the number of transitions in the SCPN). In SCPN, the infinite sequence p are all composed of m and ω. According to Proposition 3, $\exists$ a non-decreasing infinite subsequence $p' \subseteq \complement_s s^-$. In Ω_1, markings with the same tokens will be merged. Therefore, p' is increasing in the SCPN. According to the SRMG construction algorithm, when $K(j) = empty$ and $p_i < p_j$, the value of the i component is changed to ω. Similarly, at least n components of p'_n are ω, and no vector is greater than p'_n at this time, p is non-increasing subsequence. This contradicts the assumption. So the assumption is wrong and the finiteness of the SRMG is proved. □

According to the construction algorithm of the SRMG, we give the SRMG of Fig. 1a in Fig. 6. We can see that under the initial marking I_0, there are two transitions t_1 and t_2 that can be fired. The transition t_1 is a type of continuous transition, so the marking under the t_1 branch is continuous until the token in the corresponding place is consumed or consumed to the lower bound of the capacity function. And t_2 is a discrete type of transition, which transfers the token of p_1 to p_2. Therefore, the marking under the t_2 branch are all endpoints. It is worth noting that I_6 has two t_2 branch arcs pointing to it. The two arcs come from I_3 and I_5 respectively. The meaning of these two arcs is that the token in p_1 is transferred to p_2 through a discrete transition t_2. But, the source of the token in p_2 is different. Under the I_3 marking, the token value "6" in p_1 is the remaining token that is consumed after fire of continuous transition t_1, while the token value "4" in p_2 is the newly generated token after fire of continuous transition t_1. Under the I_5 marking, it's the opposite of the I_3 marking. The token value "4" in p_1 is the remaining token that is consumed after the fire of continuous transition t_1, while the token value "6" in p_2 is the newly generated token after fire of continuous transition t_1. Therefore, the two marking I_3 and I_5 are transformed into the same marking I_6 by the fire of discrete transition t_2. Under the marking I_6, the place p_2 has two different tokens, one is the newly generated token with the fire of continuous transition t_1 and the other is the initial token from p_1 with the fire of discrete transition t_2. Just because the fire sequence of continuous transition t_1 and discrete transition t_2 is different, the corresponding

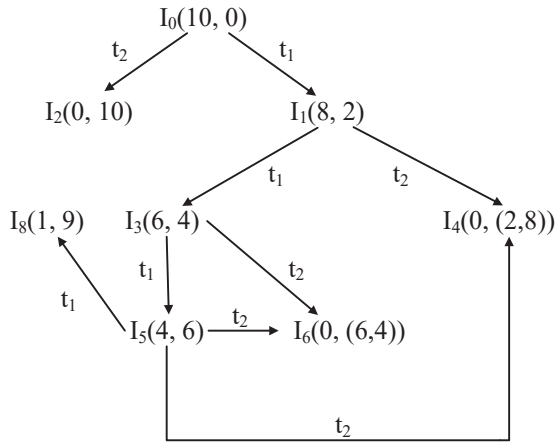

Fig. 6. SRMG of Fig. 1a.

markings are different, but they will all reach the same marking in the end. This shows that under certain conditions the order of firing of transitions will not affect the final result.

4 APPLICATION STUDY

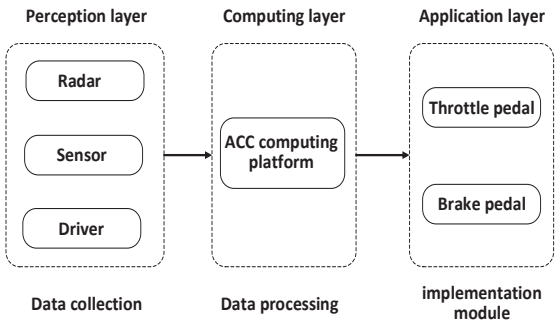

Fig. 7. The overall framk of ACC.

This section takes the modeling of ACC system as an example to introduce the modeling advantages and correctness of the SCPN. The overall control flow of the ACC system is shown in Fig. 7. The overall process is divided into three layers: perception layer, computing layer, and application layer. The perception layer obtains the input information required by the ACC system through radars and sensors that are mainly the speed of the front vehicle, the actual distance between the two vehicles, and so on. The computing layer calculates and outputs the expected following distance according to the prescribed distance algorithm. Finally, the application layer compares the actual distance with the expected distance from the vehicle and controls the throttle

pedal and brake pedal to change the speed so that the difference between them is reduced. The ACC system continuously adjusts the following distance through the above three-layer control process to complete the following driving. Therefore, we can consider the ACC system to be a closed-loop feedback control system.

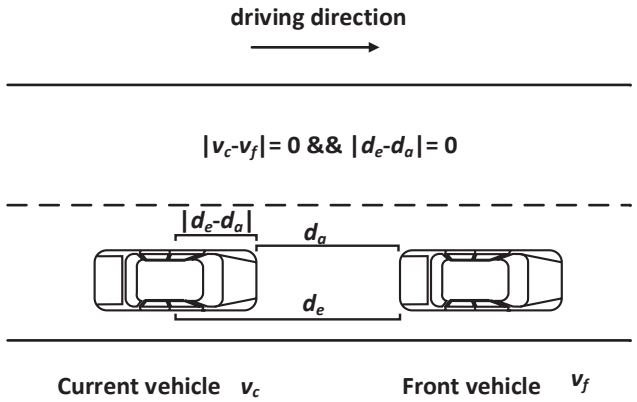

Fig. 8. Schematic diagram of following vehicle cruise.

Fig. 8 shows the process of cruising with the following front vehicle. The current vehicle needs to maintain a certain safe distance from the front vehicle in above process. If the current vehicle and the front vehicle maintain a certain safety distance, then the speed of the two vehicles must be equal. In addition, the expected distance given by the ACC system and actual distance must also be equal. The control purpose of the ACC system is to make $|d_e - d_a| = 0$ and $|v_f - v_c| = 0$ in Fig. 8 at the same time. At this time, the current vehicle has completed the synchronization with the front vehicle.

In ACC systems, the calculation of the expected distance is particularly important. The expected distance cannot be too large or too small. Expected distance is too large to cause low road utilization, while the expected distance is too small to cause traffic accidents such as rear-end collisions. Expected distance can be divided into constant expected distance and variable expected distance. Among them, due to the constancy of the constant expected distance, there are great safety hazards in the driving section where the speed of the vehicle changes greatly, and it is unable to cope with the complex and changeable driving environment. The variable expected distance introduces the time headway (the time interval required for two vehicles to pass through the same cross section), and adjusts the expected distance according to the current speed of the vehicle. When the speed of the vehicle is faster, the expected safety distance will increase accordingly to ensure the safety of the following process. when the speed of the vehicle is slow, the expected safety distance will decrease accordingly to improve the utilization rate of the traffic road, and prevent other vehicles from getting into the middle of the two vehicles. The variable expected distance is divided into the constant time headway distance and the variable time headway distance. The difference lies in whether the time headway changes with the speed of the vehicle. At present, the ACC system of most vehicles on the market adopts the constant-time-distance strategy whose algorithm is simple

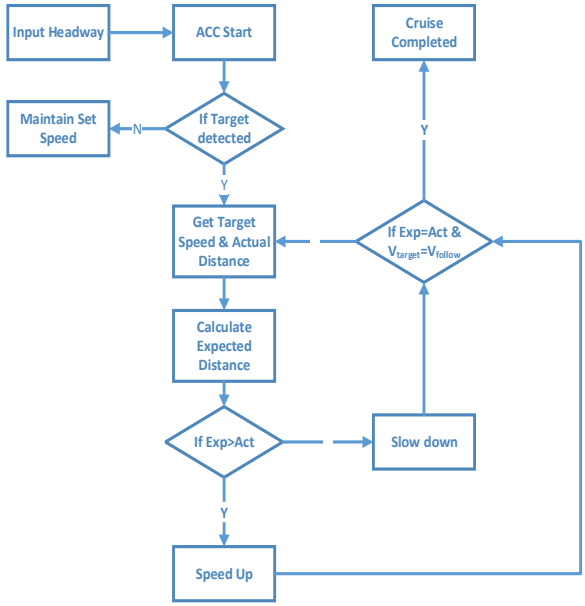

Fig. 9. Control algorithm of ACC.

and relatively mature. While the variable headway strategy is relatively imperfect. Therefore, this paper adopts the strategy of constant time headway to model the ACC.

This section mainly depicts the process from the vehicle sensor acquiring the information of the preceding vehicle to the feedback control of the ACC system. By modeling and analysing the system, the risk problems that may exist in the running of the ACC system are found and resolved to ensure maximum safety during the driving process. This paper researches the procedural modeling of these three layers. According to the relevant characteristics of SCPN and the ACC system, we can construct the SCPN model as the following modeling rules:

(1) We use the hybrid place of the SCPN to represent the speed in the ACC system. Because SCPN's hybrid place can not only participate in the firing of discrete transitions but also participate in the continuous transitions.
(2) The hybrid transition of the SCPN is used to represent the continuous quantity change and the reception processing of information in the ACC system.
(3) The discrete place of the SCPN is used for the numbering and sequential processing of information streams. Discrete places can also be used to represent the current state of the ACC system.
(4) The discrete transition of the SCPN is used to represent the changes of the ACC state.
(5) The discrete arcs of the SCPN are used to represent the input and output of the information flow in the ACC system.
(6) The continuous arc of the SCPN is used to represent the continuous change of speed and distance in the ACC system.

First of all, the speed information of the vehicle can be stored in a hybrid place. The hybrid place can be used as the front set of discrete transitions or the front set of continuous transitions. When

the hybrid place is used as the front set of discrete transitions, it means the transfer of information. When the hybrid place is used as the front set of continuous transitions, it represents the continuous change of the token in the place. The hybrid transitions are divided into two categories, one of which takes continuous quantity changes as input and discrete quantities as output. Such transitions can count the changes of continuous quantities and deal with each fire of transitions. The other type of hybrid transition is the opposite. It takes discrete quantities as input and the changes of continuous quantities as output. Such transitions are usually used to indicate updates to continuous quantities. In addition, we use discrete places to number each piece of information in the front vehicle. The purpose of doing this is to ensure that the multiple information streams transmitted by the front vehicle are processed in the order. Discrete places can also represent the current state of the ACC system. When the front discrete transition of the discrete place representing the state of the ACC system is in the enabled state, the transition is fired, and the state of the ACC system changes accordingly. Correspondingly, the discrete transitions represent the process of changing the state of the ACC system and the process of numbering the information flow of the front vehicle. Discrete arcs and continuous arcs are used according to the actual situation of the network model. Generally, a continuous arc is connected to at least one continuous transition or one continuous place. The discrete arc is connected to at least one discrete place or a discrete transition.

The control algorithm for the ACC system is given in Fig. 9. The formal model of the ACC system in this paper is based on this algorithm. Fig. 10 describes the entire running process of the ACC system, which is a simplified system model. This model mainly depicts an ACC driving scenario where the vehicle in front is continuously decelerating, which is the most prone to rear-end collision. Tab. 1 contains the representation of each symbol used in Fig. 10. The arc with a circle represents an inhibitor arc in the Petri net. If a place connected by an inhibitor arc contains a token, the associated transition cannot be triggered. A transition can only be triggered when there are no tokens in the place. For example, when the place DC receives a token and it has not yet flowed out, the transition t2 cannot be triggered. Here, we consider the actual distance between the two vehicles during the running of the ACC system. We use the capacity function to limit the actual distance between two vehicles and the speed of the two vehicles. For example, $K(AD)_{high} = 150.0$ means the maximum effective distance of the ACC system is 150 m, and the transition t_1 cannot be fired. This means that the preceding vehicle has exceeded the following distance of the ACC system, and the ACC system follow-up function is no longer applicable. $K(AD)_{low} = 5.0$ means that the minimum safety distance between the two vehicles is 5 m. This shows that the distance between the two vehicles has reached the minimum safety distance, and if the following vehicle continues to accelerate, there is a risk of rear-end collision. The transition t_1 will not be fired to avoid that the distance between the two vehicles continues to shorten.

In Fig. 10, The perception layer is represented by blue transitions and places, the computing layer is composed of red transitions and places, and the application layer is composed of green places and transitions. According to the place FCS and FCA, the speed of the front vehicle is 20 m/s, and the front vehicle is decelerating uniformly at an acceleration of 2 m/s^2. After the fire of transition t_1, each detected information stream of the front vehicle is numbered and sent to the computing layer for processing. The computing layer is mainly the fire of discrete transition t_3. The discrete place SID is to process the information flow transmitted by the perception layer in order. The discrete transition t_3 outputs the calculation result and passes it to the application layer. The application layer is based on the hybrid transition t_4 and t_5. The former controls the brake pedal to decelerate, and the latter controls the throttle pedal to accelerate. We analyze and verify the model in next section.

We now present the SRMG of the above ACC model, which is a type of reachable marking graph in Petri nets. The nodes are represented as $I_0(20, -2, 0, 13, 1, 0, 50, 0)$, indicating a marking

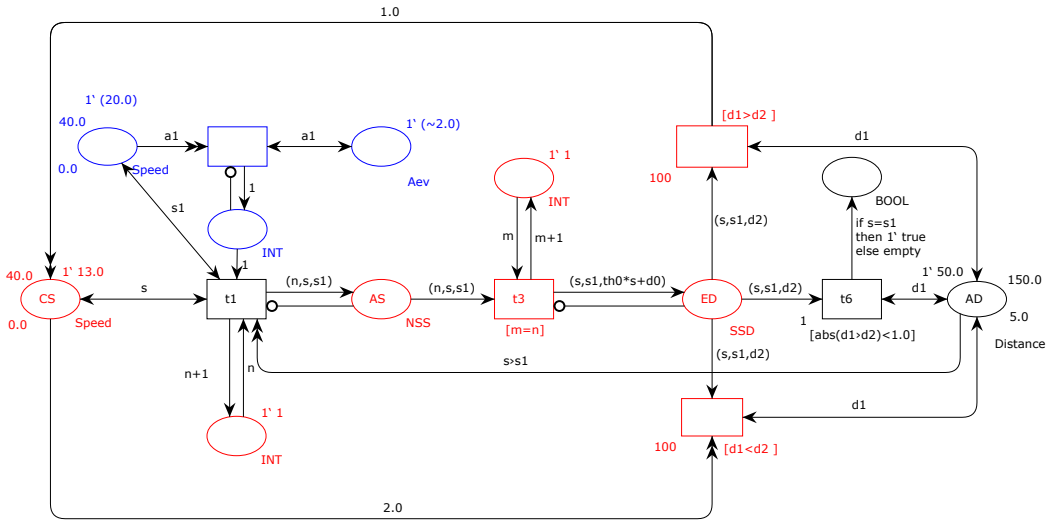

Fig. 10. SCPN of the ACC system.

Table 1. THE MEANINGS OF ELEMENTS

Element	Type	Meaning
CS	syncretic	Followed vehicle's speed
FCS	syncretic	Front vehicle's speed
FCA	discrete	Front vehicle's accelerated velocity
AS	discrete	ACC strategy
ED	discrete	Expect distance
AD	syncretic	Actually distance
DC	discrete	Data collection
SN	discrete	Number the information flow
SID	discrete	Verify and process information flow
CC	discrete	Complete the cruise
t_1	syncretic	Get the speed and distance
t_2	syncretic	Get the speed of the front vehicle
t_3	discrete	Calculation of expected distance
t_4	syncretic	Slow down
t_5	syncretic	Speed up
t_6	discrete	Completion of cruise

in the Petri net, which reflects the current state of the system. The arrows denote the transition paths from one marking to another through specific transitions, with each arrow labeled by the transition number, such as t_1, t_2, etc. These transitions signify the conditions under which the system moves from one marking to another. In general, the ACC system needs to be synchronized with the vehicle in front during driving, while avoiding rear-end collision with the front vehicle. From this perspective, this section will study the running of the ACC system in the SRMG facing the continuous deceleration of the front vehicle. In Fig. 11 the SRMG of the ACC model is composed of multiple rings, each of which represents the radar collecting the information of the front vehicle, and the flow of feedback processing by the ACC system of the following vehicle. Under the initial

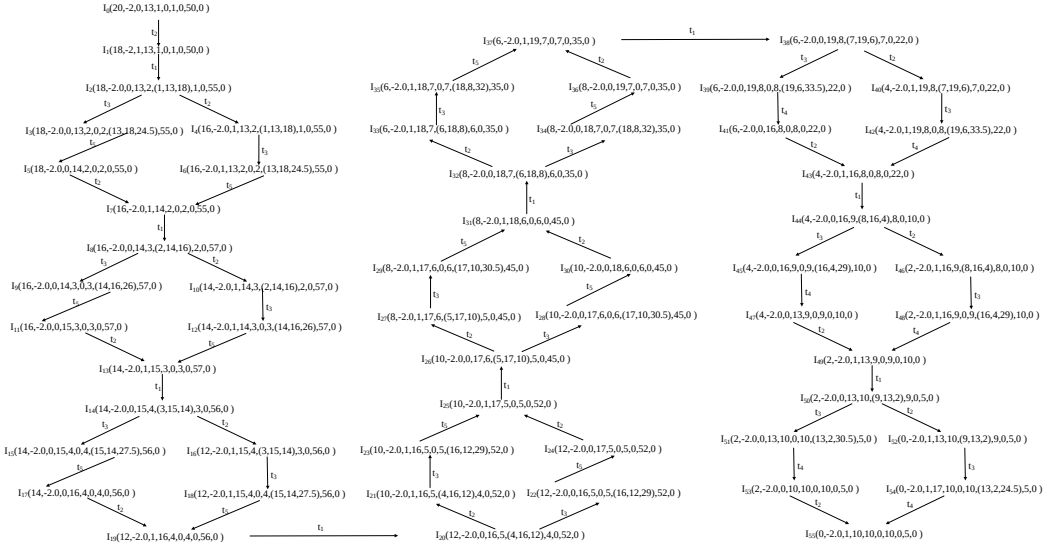

Fig. 11. SRMG of ACC.

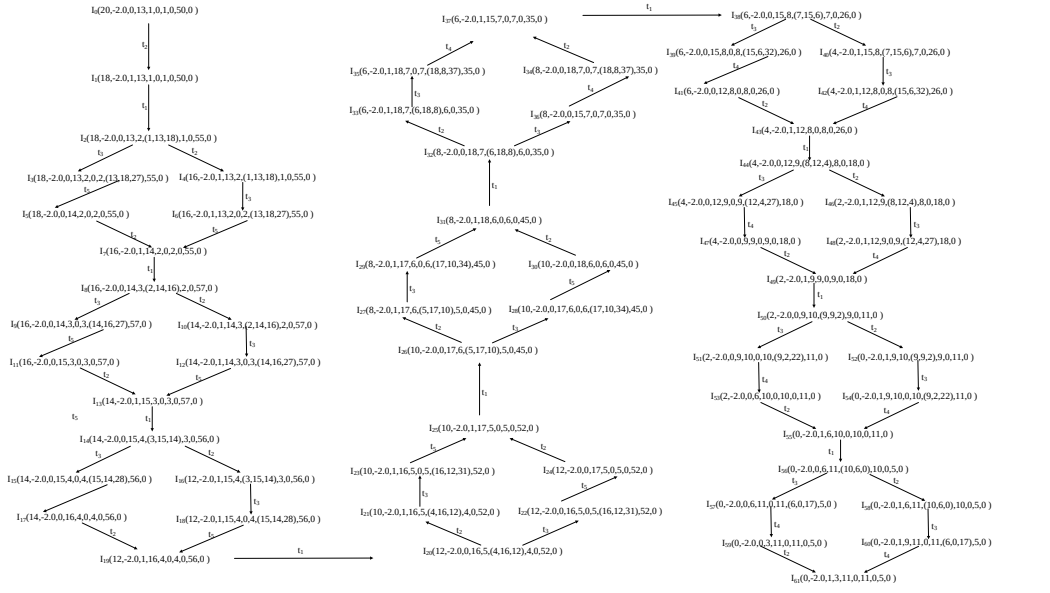

Fig. 12. SRMG of improved ACC.

marking I_0, it can be seen that the front vehicle is moving forward at a speed of 20 m/s and while is decelerating at an acceleration of 2 m/s^2 per second. The initial speed of the following vehicle is 13 m/s and the distance between the two vehicles is 50 m. According to ACC's constant headway

strategy, when the actual distance is greater than the expected distance, the following vehicle accelerates. Therefore, starting from the marking I_0, the transition t_5 is fired 6 times. This means that the following vehicle has accelerated 6 times with an acceleration of 1 m/s^2. Then the speed of the following vehicle is increased from 13 m/s to 19 m/s. At this time, the actual distance between the two vehicles is already lower than the expected following distance of the ACC system. The fire condition of transition t_4 is satisfied, and transition t_4 starts to fire. Transition t_4 was fired 3 times, and the speed of the following vehicle dropped from 19 m/s to 10 m/s. At this time, the token value in the place AD has fallen to the lower bound of the capacity function, which makes the transition t_1 unable to fire. This indicates that the distance between the two vehicles has reached the minimum safety distance, and there will be a rear-end collision risk if the ACC system continues to be used. At this time, the driver should drive the vehicle, change lanes or brake to avoid rear-end collision.

One of the typical characteristics of the SCPN is that given different initial markings, the corresponding SRMGs will be quite different. As a result of this, we conducted multiple SRMG analyses on the continuous deceleration scenarios of the preceding vehicle in different speed ranges, and the reachable marking graph obtained were all similar to those in Fig. 11. Therefore, in a scenario where the front vehicle continues to decelerate, the ACC spacing strategy will likely cause a rear-end collision risk. Through the analysis of the SRMG of the ACC model, we can find that when the front vehicle continues to decelerate, because the actual distance between the two vehicles is greater than the expected distance of the ACC system, the following vehicle will still accelerate forward. Such acceleration is unreasonable in actual driving scenes. Therefore, we make certain improvements to the ACC model. We consider adding the difference between the speeds of the two vehicles to the expected distance according to a certain proportional coefficient. When the speed of the front vehicle is much greater than the speed of the following vehicle, the risk of rear-end collision is extremely small. The relative speed of the two vehicles is negative, and the expected distance of the ACC system is the minimum safe distance, which is also in line with the ideal following distance in the actual scene. When the speed of the front vehicle is lower than the speed of the following vehicle, the risk of rear-end collision is higher. The relative speed of the two vehicles is also higher, and the expected distance of the ACC system is increased accordingly. When the front vehicle continues to decelerate, as the expected distance of the ACC system increases, the following vehicle will decelerate in advance to avoid the situation where the speed difference between the two vehicles is too large, which increases the risk of rear-end collision.

The SRMG corresponding to the improved ACC model is shown in Fig. 12. The structure of the previous part is consistent with the SRMG of the original model. After the following vehicle accelerates to 18 m/s, it can be seen from the markings I_{35} and I_{34} that the expected distance of the ACC system is greater than the actual distance between the two vehicles, so the following vehicle starts to decelerate in advance. In the SRMG of the unimproved ACC model, the transition t_4 only fired three times to reduce the speed of the following vehicle to 10 m/s until the distance between the two vehicles reached the minimum safety distance. In the SRMG of the improved ACC model, we can see that the transition t_4 fired five times to reduce the speed of the following vehicle to 3 m/s until the distance between the two vehicles reached the minimum safety distance. By comparing the termination markings of these two SRMGs, we can find that the termination state I_{55} of the unimproved SRMG indicates that the speed of the following vehicle is 10 m/s when the front vehicle stops, and the actual distance between the two vehicles has reached the minimum safety distance of 5 m. But the SRMG termination marking I_{61} of the improved ACC model indicates that the actual distance between the two vehicles has also reached the minimum safety distance of 5 m, and the improved strategy when the front vehicle stops, the speed of the following vehicle is 3 m/s. The latter is 7 m/s lower than the former, effectively reducing the risk of rear-end collision.

In addition, we also give an algorithm to output the set of dangerous markings. A marking is assumed dangerous when the velocity difference between the two vehicles is much greater than the distance, and this is guaranteed by the collision logic. Algorithm 2 is the process of searching the SRMG and finding potentially dangerous markings. The marking set Ψ that could be a potential dangerous marking set is the output of Algorithm 2.

The time complexity analysis of Algorithm 2 is crucial to understanding its efficiency, especially in large-scale systems. The algorithm begins by initializing the set Ψ and generating the SRMG under the initial marking I_0. This SRMG generation step is based on a previously defined algorithm, which has a worst-case time complexity of $O(n^2)$. Following this, an empty queue Q is defined, and if the initial marking I_0 is valid, it is added to the queue. The core of the algorithm lies in the while loop, where the complexity depends on the number of markings n processed. Each iteration of the loop involves extracting a marking from the queue, checking a condition, and computing new markings, which involves operations with a time complexity of $O(|T|)$, where $|T|$ is the number of transitions. Since the loop iterates up to n times, the total time complexity of this step is $O(n \cdot |T|)$. Therefore, the overall time complexity of **Algorithm 2** is $O(n^2 + n \cdot |T|)$.

Algorithm 2: Risk Determination Algorithm.

Input: $SCPN = (P, T, A, \Sigma, V, C, G, E, I, K)$, initial marking I_0
Output: A risky marking set Ψ
1. $\Psi = \emptyset$;
2. Get the SRMG under initial marking I_0;
3. Define an empty queue Q;
4. **if** $I_0 = none$ **then**
   Return none;
5. $Q.add(I_0)$;
6. **while** $Q \neq none$ **do**
   $I = Q.output()$;
   if $(I.CS - I.FCS) > I.AD$ **then**
   $\Psi = \Psi \cup \{I\}$;
   Calculate I' in $I]t > I'$;
   $Q.add(I')$;
   $Q = Q.next()$;

Table 2. COMPARISON OF SCPN WITH OTHER PETRI NETS

Characteristic	Semi-continuous Colored Petri Net (SCPN)	Hybrid Petri Net (HPN)	Stochastic Petri Net (SPN)	Colored Petri Net (CPN)
Description of discrete quantities	✓	✓	✓	✓
Description of continuous quantities	✓	✓	×	×
Construction of the reachable marking graph	✓	×	✓	✓
Interconversion of discrete and continuous quantities	✓	×	×	×

5 DISCUSSION

In this paper, our studies established a novel formal model and combined with a corresponding analysis method. Our study used the control of an ACC system in the scene of continuous deceleration of the front vehicle as an example to demonstrate the application of the proposed

method. The successful application of the SCPN on the ACC system showed the advantages and completeness of the SCPN for process of modeling hybrid systems. Many existing methods model discrete and continuous variations as independent components, failing to accurately capture the complexity of their interactions in real systems. Such models are often unsuitable for complex embedded control systems that need to simultaneously handle both behaviors. Traditional Petri net analysis methods, such as reachable marking graphs, face significant limitations when addressing hybrid systems, particularly in effectively analyzing continuous components. As a result, traditional methods struggle to meet the modeling requirements of complex discrete and continuous events in real industrial scenarios. Tab 2 gives a comparison of SCPN with other kinds of Petri nets. Both SCPN and HPN combine continuous and discrete quantities to model the hybrid systems. However, the HPN separately treats continuous and discrete quantities. In fact, for an actual hybrid systems, the continuous changes and overall transference of continuous quantities tend to appear in the same part. This can be depicted in Petri nets as: for the same token in a place, the continuous change and the overall transference can simultaneously occur. This is clearly not possible in the HPN. In the proposed SCPN, we weaken the continuous and discrete bounds, which is reflected in $T_D \cap T_C \neq \emptyset, P_D \cap P_C \neq \emptyset$. Therefore, the SCPN is more close to actual hybrid systems for modeling and analysing, which further improves the interpretability of the SCPN. It is worth noting that when the value of the arc function on any output continuous arc is equal to the value of the token in a place, the continuous arc seems to be equivalent to the discrete one, but the SCPN is not equivalent to the CPN. The continuous arc represents the continuous change of the token value in the place, while the discrete arc represents the transference of the token value in the place. Therefore, when the value of the arc function on any output continuous arc is equal to the value of the token in the place, the continuous arc is not equivalent to the discrete one. The proposed place that outputs continuous arcs still has a token with a value of 0 after the firing of the transition, which is essentially different from the empty place. This place can still participate in the firing of discrete transitions, and the empty place cannot participate in the firing of transitions.

The contribution of the SCPN is to replace continuous changes over a while with an average change. The advantage of this is to discretize the change of the continuous quantity to use the reachable marking graph to formally analyze the SCPN. In addition, the SCPN can also be applied to other fields which have both discrete and continuous quantities and they can be transformed into each other. The SCPN is proposed to analyze the flaws in the system process from the perspective of formal methods. Compared with traditional data analysis methods, the SCPN model is able to detect logical loopholes in the system control without relying on a large amount of data, which complements traditional data analysis methods to enhance the safety of the control system. However, the SCPN also suffers from the state explosion problem common to this type of formal method. The number of system states corresponding to the SRMG increases exponentially with the size of the system. Therefore, the combination of traditional data analysis methods and formal methods can not only enhance the safety and efficiency of the system, but also further reduce the total cost of system risk analysis.

6 CONCLUSION

This paper proposes the Semi-continuous Colored Petri Net (SCPN) as a novel formal model for hybrid system modeling and analysis. Unlike traditional methods, SCPN seamlessly integrates both discrete and continuous dynamics within the same framework. By formalizing continuous changes as average rates, it provides a more accurate representation of real-world systems. This feature enables SCPN to perform system analysis using the Reachable Marking Graph of SCPN, which is not achievable with conventional Petri net methods. The unique capability of SCPN allows it to detect logical flaws without relying on large datasets, making it a valuable complement to data-driven

approaches. Application to the Adaptive Cruise Control (ACC) system has demonstrated that the model can identify potential rear-end collision risks in complex scenarios. Future research will focus on risk analysis based on the lead vehicle's speed, acceleration, and distance, with the aim of developing new following control strategies to enhance the safety and robustness of the Adaptive Cruise Control system. Additionally, an experimental platform will be developed to further validate the feasibility of the SCPN model.

REFERENCES

[1] Francesco Basile, Pasquale Chiacchio, and Jolanda Coppola. 2011. Colored Hybrid Petri-nets for modeling material handling systems. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*. 5881–5886.

[2] Tanmay Chakraborty, Shingo Yamaguchi, Mohd Anuaruddin Bin Ahmadon, and Soumya Kanti Datta. 2019. Modeling ACC with Cloud, Clouldlet for Autonomous Vehicle Platoon using Petri nets. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. 111–116.

[3] Nivethitha Amudha Chandramohan. 2018. *Design and modeling of adaptive cruise control system using petri nets with fault tolerance capabilities*. Purdue University.

[4] Lin Chuang. 2005. *Stochastic Petri Nets and System Performance Evaluation*. Stochastic Petri Nets and System Performance Evaluation.

[5] René David and Hassane Alla. 2005. *Discrete, continuous, and hybrid Petri nets*. Vol. 1. Springer.

[6] Chanon Dechsupa, Wiwat Vatanawood, and Arthit Thongtak. 2019. Hierarchical verification for the BPMN design model using state space analysis. *IEEE Access* 7 (2019), 16795–16815.

[7] Angela Di Febbraro, Davide Giglio, and Nicola Sacco. 2016. A Deterministic and Stochastic Petri Net Model for Traffic-Responsive Signaling Control in Urban Areas. *IEEE Transactions on Intelligent Transportation Systems* 17, 2 (2016), 510–524.

[8] Reza Entezari-Maleki, Sayed Ehsan Etesami, Negar Ghorbani, Arian Akhavan Niaki, Leonel Sousa, and Ali Movaghar. 2020. Modeling and Evaluation of Service Composition in Commercial Multiclouds Using Timed Colored Petri Nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 50, 3 (2020), 947–961.

[9] Maria Pia Fanti, Giorgio Iacobellis, Agostino Marcello Mangini, and Walter Ukovich. 2014. Freeway Traffic Modeling and Control in a First-Order Hybrid Petri Net Framework. *IEEE Transactions on Automation Science and Engineering* 11, 1 (2014), 90–102.

[10] A. Favela and D. Capriles. 2017. Hybrid System Control Using Discrete State Space Analysis. *IEEE Latin America Transactions* 15, 6 (2017), 1027–1033.

[11] Zhennan Fei, Sajed Miremadi, Knut kesson, and Bengt Lennartson. 2014. Efficient Symbolic Supervisor Synthesis for Extended Finite Automata. *IEEE Transactions on Control Systems Technology* 22, 6 (2014), 2368–2375.

[12] Alessandro Giua, Stéphane Lafortune, and Carla Seatzu. 2019. Divergence properties of labeled Petri nets and their relevance for diagnosability analysis. *IEEE Trans. Automat. Control* 65, 7 (2019), 3092–3097.

[13] Carlos Gómez-Huelamo, Luis M Bergasa, Rafael Barea, Elena López-Guillén, Felipe Arango, and Pablo Sánchez. 2019. Simulating use cases for the UAH Autonomous Electric Car. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2305–2311.

[14] Haiyang Hu, Jiawei Yu, Zhongjin Li, Jie Chen, and Hua Hu. 2020. Modeling and analysis of cyber–physical system based on object-oriente generalized stochastic Petri net. *IEEE Transactions on Reliability* 70, 3 (2020), 1271–1285.

[15] Zichao Huang, Duanfeng Chu, Chaozhong Wu, and Yi He. 2019. Path Planning and Cooperative Control for Automated Vehicle Platoon Using Hybrid Automata. *IEEE Transactions on Intelligent Transportation Systems* 20, 3 (2019), 959–974.

[16] Kurt Jensen and Lars M Kristensen. 2009. *Coloured Petri nets: modelling and validation of concurrent systems*. Springer Science & Business Media.

[17] Lei Jiang, EnLiang Liu, Ding Lui, and Jia Zhai. 2018. Modeling and Control of BUCK Circuit Based on Hybrid Automata. In *2018 Chinese Automation Congress (CAC)*. 2499–2502.

[18] Yan-Fu Li, Enrico Zio, and Yan-Hui Lin. 2012. A Multistate Physics Model of Component Degradation Based on Stochastic Petri Nets and Simulation. *IEEE Transactions on Reliability* 61, 4 (2012), 921–931.

[19] Cong Liu and Jie Wu. 2013. Fast Deep Packet Inspection with a Dual Finite Automata. *IEEE Trans. Comput.* 62, 2 (2013), 310–321.

[20] Xiao-li Lu, Yun-wei Dong, Bo Sun, and Hong-bin Zhao. 2011. Research of embedded software testing method based on AADL modes. In *2011 IEEE 3rd International Conference on Communication Software and Networks*. IEEE, 89–92.

[21] Ziyue Ma, Guanghui Zhu, Zhiwu Li, and Alessandro Giua. 2019. Computation of admissible marking sets in weighted synchronization-free Petri nets by dynamic programming. *IEEE Trans. Automat. Control* 65, 6 (2019), 2662–2669.

[22] Arash Mahboubi, Seyit Camtepe, and Hasmukh Morarji. 2017. A Study on Formal Methods to Generalize Heterogeneous Mobile Malware Propagation and Their Impacts. *IEEE Access* 5 (2017), 27740–27756.

[23] Negar Majma and Seyed Morteza Babamir. 2020. Model-Based Monitoring and Adaptation of Pacemaker Behavior Using Hierarchical Fuzzy Colored Petri-Nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 50, 9 (2020), 3344–3357.

[24] Karima Outafraout, Ahmed Nait-Sidi-Moh, and El Houcine Chakir El Alaoui. 2020. A Control Approach Based on Colored Hybrid Petri Nets and (Max, +) Algebra: Application to Multimodal Transportation Systems. *IEEE Transactions on Automation Science and Engineering* 17, 3 (2020), 1208–1220.

[25] André Platzer. 2010. Differential-algebraic dynamic logic for differential-algebraic programs. *Journal of Logic and Computation* 20, 1 (2010), 309–352.

[26] Ricardo J. Rodr¨a guez, Simona Bernardi, and Armin Zimmermann. 2020. An Evaluation Framework for Comparative Analysis of Generalized Stochastic Petri Net Simulation Techniques. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 50, 8 (2020), 2834–2844.

[27] Tomasz Szmuc and Wojciech Szmuc. 2020. Consistency Preserving Development of Embedded Systems Using AADL. In *2020 27th International Conference on Mixed Design of Integrated Circuits and System (MIXDES)*. 245–248.

[28] Carlos R Vazquez and Manuel Silva. 2015. Stochastic hybrid approximations of Markovian Petri nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45, 9 (2015), 1231–1244.

[29] Bohan Wang, Wenjun Ke, Jianwei Zhang, Xinrui Gao, Jing Chen, Kunlong Wang, Yuting Yang, and Yifei Da. 2018. A Method of Software System Security Verification and Evaluation Based on Extension of AADL Model. In *2018 Eighth International Conference on Instrumentation Measurement, Computer, Communication and Control (IMCCC)*. 1726–1731.

[30] Xiaomin Wei. 2019. AADL-Based Safety Analysis Approaches for Safety-Critical Systems. In *2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)*. 481–482.

[31] WangYang Yu, ChunGang Yan, ZhiJun Ding, ChangJun Jiang, and MengChu Zhou. 2014. Modeling and Validating E-Commerce Business Process Based on Petri Nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 44, 3 (2014), 327–341.

[32] Wangyang Yu, Chungang Yan, Zhijun Ding, Changjun Jiang, and Mengchu Zhou. 2018. Analyzing E-Commerce Business Process Nets via Incidence Matrix and Reduction. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 48, 1 (2018), 130–141.

[33] WangYang Yu, Chun Gang Yan, ZhiJun Ding, ChangJun Jiang, and MengChu Zhou. 2016. Modeling and Verification of Online Shopping Business Processes by Considering Malicious Behavior Patterns. *IEEE Transactions on Automation Science and Engineering* 13, 2 (2016), 647–662.

[34] Cheng Yuehua, Jiang Liang, Jiang Bin, and Lu Ningyun. 2019. Useful life prediction using a stochastic hybrid automata model for an ACS multi-gyro subsystem. *Journal of Systems Engineering and Electronics* 30, 1 (2019), 154–166.

[35] Zhen Zhao, Jun Zhang, Yigang Sun, and Zhexu Liu. 2018. Modeling of avionic display system for civil aircraft based on AADL. In *2018 Chinese Control And Decision Conference (CCDC)*. 4121–4126.

[36] Wu Zhehui. 2006. *Introduction to Petri Nets.* Introduction to Petri Nets.