Compositional Time Petri Nets and Reduction Rules

Jiacun Wang, Yi Deng, *Member, IEEE*, and MengChu Zhou, *Senior Member, IEEE*

***Abstract*—This paper introduces compositional time Petri net (CTPN) models. A CTPN is a modularized time Petri net (TPN), which is composed of components and connectors. The paper also proposes a set of component-level reduction rules for TPNs. Each of these reduction rules transforms a TPN component to a very simple one while maintaining the net's external observable timing properties. Consequently, the proposed method works at a coarse level rather than at an individual transition level. Therefore, one requires significantly fewer applications to reduce the size of the TPN under analysis than those existing ones for TPNs. The use and benefits of CTPNs and reduction rules are illustrated by modeling and analyzing the response time of a command and control system to its external arriving messages.**

***Index Terms*—Compositional modeling, formal verification, reduction, system analysis, time Petri nets.**

I. Introduction

PETRI NETS [6], [7] have been used to model various discrete event systems [17], [20]. Because of their ability to model asynchronous events, parallelism, contention, and synchronization, they have gained more and more applications. Basic Petri nets lack a temporal description and, therefore, fail to represent any timing constraints for time-dependent systems. Introduction of time into a transition, place, or arc increased both the modeling power and the complexity of the net analysis. Several extended models of Petri nets were proposed to deal with the timing issues [12]. These models include timed Petri nets [19], stochastic timed Petri nets [4], and time Petri nets (TPNs) [5]. Among these models, TPNs are most widely used for real-time system specification and verification [3], [13], [14], [10]. In TPNs, event synchronization is represented in terms of a set of pre- and post-conditions associated with each individual action of the modeled system, and timing constraints are expressed in terms of minimum and maximum amount of time elapsed between the enabling and the execution of each action. This allows a compact representation of the state space and an explicit modeling of concurrency and parallelism.

A fundamental and most widely applied method for analyzing TPNs, as for many other formal models, is reachability analysis [2], [1], [15]. It permits the automatic translation of behavioral specification models into a state transition graph made up of a set of states, a set of actions, and a succession relation associating states through actions [3]. This representation makes explicit such properties as deadlock and reachability [18], and allows the automatic verification of ordering relationships among task execution times [10].

However, for a complex or even midsize TPN, it is difficult to enumerate its reachable states, which is commonly referred to as a state-explosion problem. Sloan *et al.* developed several reduction rules for TPN analysis that work at an individual transition level [8]. These reduction rules help to reduce the complexity of TPN analysis to some extent. However, it is not a trivial work to automatically search the preconditions of applying these reduction rules for a complex TPN.

Modern complex time-dependent systems are often of module constructs. A modular system is a composition of components that interact with each other through connectors. In this paper, we introduce the concept of compositional time Petri nets (CTPNs). A CTPN is a modularized TPN, which is composed of components and connectors. A component is a coarse grained subnet of a TPN, and a connector is a simple TPN to describe the interaction among components. We propose a set of component-level reduction rules for TPNs. Each of the reduction rules transforms a TPN component to a small one while maintaining the net's external observable timing properties. The application of these rules will dramatically reduce the size of a CTPN. Meanwhile, because these rules work at a much coarser level than those developed by Sloan *et al.*, fewer applications of our rules are needed to reduce the size of the TPN under consideration.

Many research achievements have been reported in reduction techniques for general Petri nets. In particular, a stepwise refinement and abstraction method [9], [11] is developed for Petri nets, where the abstraction (reduction) technique can be used as a "divide-and-conquer" approach for the analysis of liveness, boundness, resource requirements, etc., for complex Petri nets. Our reduction rules are similar in spirit to their work [9], [11], in that we reduce the TPN components to the same form of simple Petri nets. The difference is that we reduce TPN components in terms of equivalent external observable timing properties. The analysis of TPN is clearly more challenging and important since it can model and reveal more characteristics of a discrete event system that evolves over time.

The rest of the paper is arranged as follows: Section II introduces the concept of CTPN models. Section III proposes a set of component-level reduction rules. Section IV illustrates the use and benefits of CTPNs and reduction rules by modeling and analyzing the response time of a command and control system to its external arriving message. Section V makes concluding remarks.

Manuscript received October 29, 1999; revised May 28, 2000. This work was supported in part by the Army Research Office under Grant DAAG55-98-1-0428 and by the National Science Foundation under Grant HDR-9707076. This paper was recommended by Associate Editor Y. Narahari.

J. Wang is with the School of Computer Science, Florida International University, Miami, FL 33199 USA.

Y. Deng was with the School of Computer Science, Florida International University, Miami, FL 33199 USA. He is now with the Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083 USA.

M. Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07012 USA (e-mail: zhou@njit.edu).

Publisher Item Identifier S 1083-4419(00)06716-9.

II. Compositional Time Petri Nets

CTPNs augment the basic Petri net model with timing constraints of TPNs and modular constructs. They follow the same transition firing rules as TPNs [5].

A. Petri Nets

A Petri net is a bipartite directed graph in which the nodes are called *places* and *transitions*. A Petri net is a 4-tuple (P,T,I,O), where P is a set of places ($|P| = m$); $T$is a set of transitions ($|T| = n$, $P \cap T = \emptyset$, $P \cup T \neq \emptyset$); and $I(O)$ is the pre- (post-) incidence function representing the input (output) arcs $I : P \times T \to N = \{0,1,2,\ldots\} (O : P \times T \to N)$.

The pre- and post-sets of a transition $t \in T$ are defined as ${}^\bullet t = \{p \,|\, I(p,t) > 0\}$ and $t^\bullet = \{p \,|\, O(p,t) > 0\}$, where $I(p,t)$ and $O(p,t)$ are the multiplicities of arcs (p,t) and (t,p), respectively. The pre- and post-sets of place $p \in P$ are defined as ${}^\bullet p = \{t \,|\, O(p,t) > 0\}$ and $p^\bullet = \{t \,|\, I(p,t) > 0\}$, respectively.

A place may have *tokens*. A function $M : P \to N$ is called a *marking*, which is usually represented as a column vector where each element is the number of tokens contained in the corresponding place. A marking represents a (distributed) state of the modeled system. A *marked* Petri net (P,T,I,O,M_0) is a Petri net with an initial marking M_0. A transition $t \in T$ is enabled in M iff $M(p) \geq I(p,t)$ for any $p \in P$. A transition t enabled in M can *fire* and, thus, yield a new marking $M'(p) = M(p) - I(p,t) + O(p,t)$ for any $p \in P$.

B. Time Petri Nets

Time Petri nets (TPNs) were first introduced by Merlin and Farber [5]. In a TPN, two time values are defined for each transition t, $EFT(t)$, and $LFT(t)$, where $EFT(t)$ is the minimum time the transition must wait after it is enabled and before it is fired, i.e., its *earliest firing time*, and $LFT(t)$ the maximum time the transition can wait before firing if it is still enabled, i.e., its *latest firing time*. Times $EFT(t)$ and $LFT(t)$ are relative to the moment at which t is enabled. Assume that t has been enabled at global time τ. Even though it is continuously enabled, it cannot fire before $\tau + EFT(t)$, and must fire before or at time $\tau + LFT(t)$, unless it is disabled before firing due to another transition's firing. Formally, a TPN is a 6-tuple (P,T,I,O,M_0,SI), where (P,T,I,O,M_0) is a marked Petri net and SI is a mapping called *static interval*, $SI : T \to Q^\bullet \times (Q^\bullet \cup \infty)$, where $Q^\bullet$ is the set of nonnegative rational numbers.

A *state* S of a TPN is a pair $S = (M,F)$, where M is a marking and F a firing interval set which is a vector of possible firing times. The number of entries in this vector is given in the number of the transitions enabled by marking M.

Suppose that transition t_0 fires at global time θ_0 at state $S_0 = (M_0,F_0)$ and results in state $S_1 = (M_1,F_1)$, t_1 firing at global time θ_1 at state S_1 and results in state $S_2 = (M_2,F_2),\ldots,$ and t_i firing at global time θ_i at state S_i and results in state $S_{i+1} = (M_{i+1},F_{i+1})$. Then we get a *firing schedule* $\omega = (t_0,\theta_0)(t_1,\theta_1)\cdots(t_i,\theta_i)$. In general, for TPN E with schedule ω, we denote by $\varphi(E,\omega)$ the state reached by starting in E's initial state and firing each transition in ω at its associated time, and $\theta(E,\omega)$ the global time when state $\varphi(E,\omega)$ is reached, which is the global firing time of the last transition in ω. We also denote by $L(E)$ the set of all firing schedules of E.

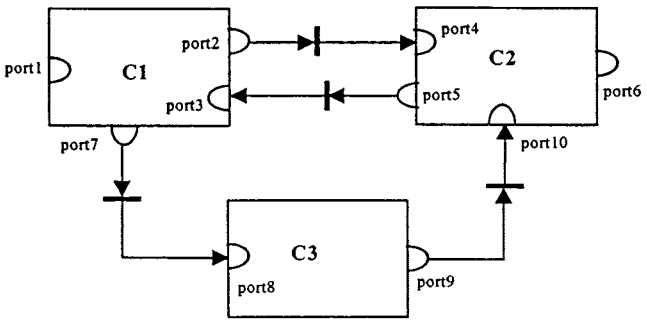


Fig. 1. Illustration of an example CTPN structure.

C. Compositional Time Petri Nets

The building blocks of a CTPN are *components* [14]. A component is a coarse grained subnet of a TPN. A CTPN model consists of two basic elements: *components* and *connectors*. Connectors are also represented by TPN models. The component TPN models describe the real-time behavior and communication interface of their corresponding components. The connectors specify how the components interact with each other. Both are used to form a composition model. All connectors are defined using only communication interfaces, which gives us the flexibility to change the design of individual components without a need to void the analysis of the entire system.

Fig. 1 shows an example structure of CTPN models. It has three components—$C1, C2$, and $C3$. Each component model has two parts: 1) *communication ports* (denoted graphically by half circles), including *input ports* (e.g., *port*8) and *output ports* (e.g., *port*9) and 2) a TPN that describes the time-dependent operational behavior of the component, i.e., it defines the semantics associated with the ports. The communication between a component and its environment is solely through the ports. A connector represents a channel of interaction between components. It is modeled by a simple TPN and defines the direction of message flow and delay in the channel. For example, components $C1$ and $C2$ have a request–reply relationship that is modeled by a bidirectional channel. The communication is asynchronous message passing.

Definition 1: Let $E = (P,T,I,O,M_0,SI)$ be the time Petri net model of a component, and

$$\begin{aligned} PIN &= \{p \,|\, p \in P, {}^\bullet p \cap T = \emptyset\} \\ POUT &= \{p \,|\, p \in P, p^\bullet \cap T = \emptyset\} \\ PORT &= PIN \cup POUT \\ &= \{p \,|\, p \in P, {}^\bullet p \cap T = \emptyset \vee p^\bullet \cap T = \emptyset\}. \end{aligned}$$

Then, $p \in PIN$ is called an *input port*, $p \in POUT$ is called an *output port*, and $p \in PORT$ is called a *port* of the component.

III. Component-Level Reduction Rules

In this section, we present a set of component-level reduction rules for TPNs. These rules preserve the external observ-

able timing properties of the components to be reduced, and are fundamentals to the analysis of CTPN models.

Following [8], we use $E \mid X$ for the time Petri net obtained by restricting the underlying graph to X. This notation is also used for net states and firing schedules. For time Petri net $E = (P, T, I, O, M_0, SI)$ and $X \subset T$, we define $L(E) \mid X$ as the set of all schedules of E with all pairs containing a transition not from X deleted. $L(E) - X = L(E) \mid (T \backslash X)$, where $T \backslash X$ is the set of transitions in T but not in $T \cap X$. $E - \{t1, t2\}$ refers to what is left on net E when one removes transitions $t1$ and $t2$ together with all arcs incident on either $t1$ or $t2$. If (M, F) is a state of net E, then $(M, F) - \{t, p\}$ is the same state with place p deleted from the marking and transition t's firing interval, if any, deleted from vector F.

Definition 2: Assume that a rule transforms a time Petri net E to E'. Let U be the set of transitions of E which are left completely unmodified by this transformation. We say that the transformation is *timing property preserving* if $L(E) \mid U = L(E') \mid U$.

Based on this definition, when we reduce a component in a CTPN into a simple TPN using a timing property preserving transformation rule, the timing property, in terms of transition firing schedules, for the rest of the CTPN remains unchanged.

In the rest of this section, we will present several reduction rules that fuse together two or more transitions. We will give the proof that the first rule is timing property preserving in complete detail. In addition, the following interval arithmetic is used. Let $F_1 = [a_1, b_1]$ and $F_2 = [a_2, b_2]$, with $0 \leq a_i \leq b_i \leq +\infty$. Then we define $F_1 + F_2$ to be the interval $[a_1 + a_2, b_1 + b_2]$. In the special case where F_2 is a point interval, i.e., $a_2 = b_2$, and $a_2 \leq a_1$, we define $F_1 - F_2$ to be the interval $[a_1 - a_2, b_1 - b_2]$.

The following definition is useful in the proofs of the following theorems.

Definition 3: We say that a port p gets marked *during* a schedule ω if there is a prefix ω_1 of ω such that p is marked in the state $\varphi(E, \omega_1)$.

Component-Level Reduction Rule 1: Let E be the TPN model of a system, and E_C be the TPN model of component C in the system, with $C.PIN = \{port1\}$ and $C.POUT = \{port2\}$. The component has no enabled transition under the initial marking of E. If

1) whenever *port*1 receives a token, *port*2 is guaranteed to receive a token in the future;
2) *port*1 cannot receive another token until *port*2 has received a token

then we can reduce E into E' by replacing E_C with a simple net composed of two places: *port*1 and *port*2, and one transition: t, such that

1) $port1^\bullet = {}^\bullet port2 = \{t\}, {}^\bullet t = \{port1\}, t^\bullet = \{port2\}$, while ${}^\bullet port1$ and $port2^\bullet$ remain unchanged;
2) $SI(t) = SI\_(port1, port2)$, where $SI\_(port1, port2)$ is the time delay interval for the token moving from *port*1 to *port*2 (see Fig. 2).

Theorem 1: The component-level reduction rule 1 is timing property preserving.

Proof: See the Appendix. □

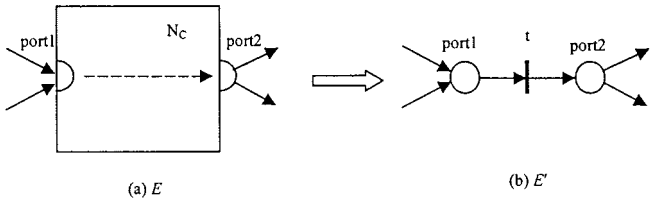


Fig. 2. Illustration of component-level reduction rule 1.

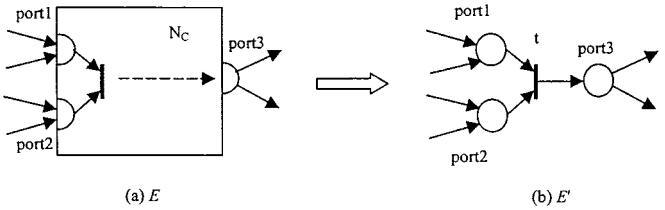


Fig. 3. Illustration of component-level reduction rule 2.

Component-Level Reduction Rule 2: Let E be the TPN model of system S, and E_C the TPN model of component C of S, with $C.PIN = \{port1, port2\}, C.POUT = \{port3\}$, and $port1^\bullet = port2^\bullet$. The component has no enabled transition under the initial marking of E. If

1) whenever both *port*1 and *port*2 receive a token, *port*3 is guaranteed to receive a token in the future;
2) at least one of *port*1 and *port*2 cannot receive another token until *port*3 has received a token

then we can reduce E into E' by replacing N_C with a simple net composed of three places: *port*1, *port*2 and *port*3, and one transition: t, such that

1) $port1^\bullet = port2^\bullet = {}^\bullet port3 = \{t\}, {}^\bullet t = \{port1, port2\}, t^\bullet = \{port3\}$, while ${}^\bullet port1, {}^\bullet port2$ and $port3^\bullet$ remain unchanged;
2) $SI(t) = SI\_((port1, port2), port3)$, where $SI\_((port1, port2), port3)$ is the time delay interval from two tokens arriving in *port*1 and *port*2 to a token reaching *port*3 (see Fig. 3).

Theorem 2: The component-level reduction rule 2 is timing property preserving.

Proof: Since $port1^\bullet = port2^\bullet$, *port*1 and *port*2 share the same output transitions, which implies that only when both *port*1 and *port*2 get tokens can some transition in $L(E) - E_C.T$ be enabled. Therefore, we can prove this theorem using the same method as that used in proving Theorem 1. □

Reduction rule 2 can be extended to cases with more than two input ports.

Component-Level Reduction Rule 3: Let E be the TPN model of a system S, and E_C the TPN model of component C of S, with $C.PIN = \{port1\}$ and $C.POUT = \{port2, port3\}$. The component has no enabled transition under the initial marking of E. If

1) whenever *port*1 receives a token, one and only one of *port*2 and *port*3 is guaranteed to receive a token in the future;
2) *port*1 cannot receive another token until one of *port*2 and *port*3 has received a token

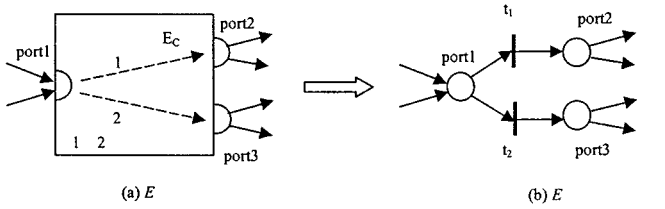


Fig. 4. Illustration of component-level reduction rule 3: The token reaching *port*1 will follow either token flow path 1 to reach *port*2, or token flow path 2 to reach *port*3.

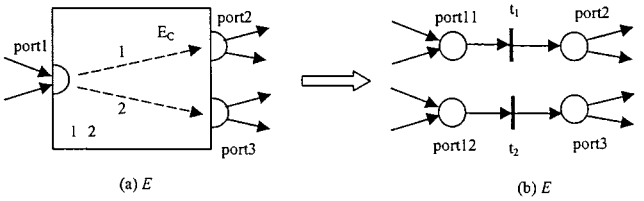


Fig. 5. Illustration of component-level reduction rule 4. The arrival of a token at *port*1 will result in a token reaching both *port*2 and *port*3.

then we can reduce E into E' by replacing E_C with a simple net composed of three places: *port*1, *port*2 and *port*3, and two transition: t_1 and t_2, such that

1) $port1^\bullet = \{t_1, t_2\}, {}^\bullet port2 = \{t_1\}, {}^\bullet port3 = \{t_2\}, {}^\bullet t_1 = {}^\bullet t_2 = \{port1\}, t_1^\bullet = \{port2\}, t_2^\bullet = \{port3\}$, while ${}^\bullet port1, port2^\bullet$ and $port3^\bullet$ remain unchanged;
2) $SI(t_1) = SI\_(port1, port2)$, and $SI(t_2) = SI\_(port1, port3)$ (see Fig. 4).

Theorem 3: The component-level reduction rule 3 is timing property preserving.

Proof Sketch: Let time Petri net E' be derived from E by applying the component reduction rule 3. The general skeleton of the proof is the same as the proof of Theorem 1. We need to prove that $L(E) - E_C.T \subseteq L(E') - \{t_1, t_2\}$ and $L(E') - \{t_1, t_2\} \subseteq L(E) - E_C.T$. We first prove $L(E) - E_C.T \subseteq L(E') - \{t_1, t_2\}$. We will show that if $\omega' \in L(E) - N_C.T$, then $\omega' \in L(E') - \{t_1, t_2\}$. Because any such ω' can be expressed as $\omega - E_C.T$ where $\omega \in L(E)$, it suffices to show that for any $\omega \in L(E), \omega' \in L(E') - \{t_1, t_2\}$ where $\omega' = \omega - E_C.T$.

We break our proof into several cases.

Case 1) *Port*1 does not get marked during ω.

Case 2) *Port*1 gets marked but neither *port*2 nor *port*3 gets marked during ω.

From the proof of Theorem 1 we can conclude $\omega' \in L(E') - \{t_1, t_2\}$ in the above two cases.

Case 3) *Port*1 gets marked once and only once, and either *port*2 or *port*3 gets marked once and only once during ω.

Suppose that *port*1 gets marked during ω. Then it follows from the proof of Theorem 1 that $\omega' \in L(E') - \{t_1, t_2\}$. If the case is that *port*3 gets marked instead of *port*2 during ω, we symmetrically have the same conclusion.

Case 4) General case.

Suppose that *port*1 gets marked a times, *port*2 gets marked b_1 times, and *port*3 gets marked b_2 times during ω. It follows from precondition (2) of this rule that either $a = b_1 + b_2$ or $a = b_1 + b_2 + 1$. We have proven the cases in which $a = 0, (a = 1) \wedge (b_1 + b_2 = 0)$, and $(a = 1) \wedge (b_1 + b_2 = 1)$. By the recursive use of the proofs for Case 2 and Case 3, we can prove that $\omega' \in L(E') - \{t_1, t_2\}$ for any schedule ω in which $b_1 + b_2$ takes any positive integral value.

In the other direction, we may show $L(E') - \{t_1, t_2\} \subseteq L(E) - E_C.T$ using the method similar to that used in proving Theorem 1. □

Reduction rule 3 can be extended to cases with more than two output ports.

Component-Level Reduction Rule 4: Let E be the TPN model of a system S, and E_C be the TPN model of component C of S, with $C.PIN = \{port1\}$ and $C.POUT = \{port2, port3\}$. The component has no enabled transition under the initial marking of E. If

1) whenever *port*1 receives a token, both *port*2 and *port*3 are guaranteed to receive a token in the future;
2) *port*1 cannot receive another token until both *port*2 and *port*3 have received a token

then we can reduce E into E' by replacing E_C with a simple net composed of four places: *port*11, *port*12, *port*2 and *port*3, and two transitions: t_1 and t_2, such that

1) ${}^\bullet port11 = {}^\bullet port12 = {}^\bullet port1, port11^\bullet = \{t_1\}, port12^\bullet = \{t_2\}, {}^\bullet t_1 = \{port11\}, {}^\bullet t_2 = \{port12\}, t_1^\bullet = \{port2\}, t_2^\bullet = \{port3\}$, while $port2^\bullet$ and $port3^\bullet$ remain unchanged;
2) $SI(t_1) = SI\_(port1, port2)$, and $SI(t_2) = SI\_(port1, port3)$ (see Fig. 5).

Theorem 4: The component-level reduction rule 4 is timing property preserving.

Proof Sketch: Let time Petri net E' be derived from E by applying the component reduction rule 4. The general skeleton of the proof is the same as the proof of Theorem 1. We need to prove both $L(E) - N_C.T \subseteq L(E') - \{t_1, t_2\}$ and $L(E') - \{t_1, t_2\} \subseteq L(E) - E_C.T$. We first prove $L(E) - E_C.T \subseteq L(E') - \{t_1, t_2\}$. We will show that if $\omega' \in L(E) - E_C.T$, then $\omega' \in L(E') - \{t_1, t_2\}$. Because any such ω' can be expressed as $\omega - E_C.T$ where $\omega \in L(E)$, it suffices to show that for any $\omega \in L(E), \omega' \in L(E') - \{t_1, t_2\}$ where $\omega' = \omega - E_C.T$.

We break our proof into several cases:

Case 1) *Port*1 doesn't get marked during ω.

Case 2) *Port*1 gets marked but neither *port*2 nor *port*3 gets marked during ω.

From the proof of Theorem 1 we can conclude $\omega' \in L(E') - \{t_1, t_2\}$ in the above two cases.

Case 3) *Port*1 gets marked once and only once, and either *port*2 or *port*3 gets marked during ω.

Suppose that *port*2 gets marked during ω. Then it follows from the proof of Theorem 1 that $\omega' \in L(E') - \{t_1, t_2\}$. If the case is that *port*3 gets marked instead of *port*2 during ω, we symmetrically have the same conclusion.

Case 4) *Port*1 gets marked once and only once, and both *port*2 and *port*3 get marked during ω.

Without loss of generality, we assume that *port*2 gets marked first and then *port*3 gets marked. Let ω_1 be the shortest prefix of ω such that *port*1 gets

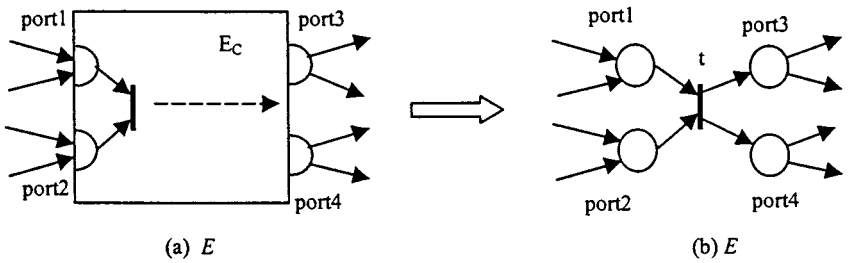


Fig. 6. Illustration of component reduction rule 5.

marked during ω. This implies that $\omega_1 - \{t_1\} = \omega_1, \omega_1\omega_{21}$ is the shortest prefix of ω such that *port*2 gets marked during ω, and $\omega_1\omega_{21}\omega_{22}$ is the shortest prefix of ω such that *port*3 gets marked during ω. Then we write $\omega = \omega_1\omega_{21}\omega_{22}\omega_3$, where $\omega_3 - \{t_1\} = \omega_3$ since we have assumed that *port*1 gets marked only once during ω.

Let θ_1 be the time of state $\varphi(E, \omega_1), \theta_{21}$ the time of $\varphi(E, \omega_1\omega_{21})$, and θ_{22} the time of $\varphi(E, \omega_1\omega_{21}\omega_{22})$. Then, from preconditions (1) and (2) of this rule, we know $\theta_{22} - \theta_1$ represents the time delay the message transfer between *port*1 and *port*3, i.e., $\theta_{22} - \theta_1 \in SI\_(port1, port3), \max(\theta_{22} - \theta_1) = \sup(SI\_(port1, port3))$, and $\min(\theta_{22} - \theta_1) = \inf(SI\_(port1, port3))$. Since $SI(t_2) = SI\_(port1, port3)$, we have $\theta_{22} - \theta_1 \in SI(t_2), \max(\theta_{22} - \theta_1) = \sup(SI(t_2))$, and $\min(\theta_{22} - \theta_1) = \inf(SI(t_2))$. Also, it follows from $\omega_1 - E_C.T = \omega_1$ and $E - E_C = E' - \{port1, port2, port3, t_2\}$ that $\omega_1 \in L(E')$ and t_2 is enabled under state $\varphi(E', \omega_1)$. In Case 3), we already have $\omega_1(\omega_{21} - E_C.T)(t_1, \theta_{21})(t_2, \theta_{22}) \in L(E')$; now we have $\omega_1(\omega_{21} - E_C.T)(t_1, \theta_{21}) \in L(E')$, and again because $E - E_C = E' - \{port11, port12, port2, port3, t_1, t_2\}$, the state $\varphi(E', \omega_1(\omega_{21} - E_C.T)(t_1, \theta_{21})(t_2, \theta_{22}))$ is followed by ω_3. It follows from $\omega_3 - E_C.T = \omega_3$ that $\omega_1(\omega_{21} - E_C.T)(t_1, \theta_{21})(t_2, \theta_{22})\omega_3 \in L(E')$, or $\omega_1(\omega_{21} - E_C.T)\omega_{22} \in L(E') - \{t_1, t_2\}$, or $\omega' \in L(E') - \{t_1, t_2\}$.

If the case is that *port*3 gets marked first and then *port*2 gets marked during ω, we symmetrically have the same conclusion.

Case 5) General case.

Suppose that *port*1 gets marked a times, *port*2 gets marked b_1 times, and *port*3 gets marked b_2 times during ω. It follows from precondition (2) of this rule that either $b_1 = a$ or $b_1 = a - 1, b_2 = a$ or $b_2 = a - 1$. We have proven the cases in which $a = 0, (a = 1) \wedge (b_1 = 0) \wedge (b_2 = 0)$, and $(a = 1) \wedge ((b_1 = 1) \vee (b_2 = 1))$. By the recursive use of the proofs for Case 2, Case 3, and Case 4, we can prove that $\omega' \in L(E') - \{t_1, t_2\}$ for any schedule ω in which a takes any positive integral value.

In the other direction, we can show $L(E') - \{t_1, t_2\} \subseteq L(E) - E_C.T$ using the method similar to that used in proving Theorem 1. □

Reduction rule 4 can be extended to cases with more than two output ports.

Component-Level Reduction Rule 5: Let E be the TPN model of a system S, and E_C be the TPN model of component C of S, with $C.PIN = \{port1, port2\}, C.POUT = \{port3, port4\}$, and $port1^\bullet = port2^\bullet$. The component has no enabled transition under the initial marking of E. If

1) whenever both *port*1 and *port*2 receive a token, both *port*3 and *port*4 are guaranteed to receive a token simultaneously in the future and
2) at least one of *port*1 and *port*2 cannot receive another token until both *port*3 and *port*4 have received a token

then we can reduce E into E' by replacing E_C with a simple net composed of four places: *port*1, *port*2, *port*3 and *port*4, and one transition: t, such that

1) $port1^\bullet = port2^\bullet = {}^\bullet port3 = {}^\bullet port4 = \{t\}, {}^\bullet t = \{port1, port2\}$ and $t^\bullet = \{port3, port4\}$, while ${}^\bullet port1$, ${}^\bullet port2, port3^\bullet$ and $port4^\bullet$ remain unchanged;
2) $SI(t_1) = SI\_((port1, port2), (port3, port4))$ [see Fig. 6].

Theorem 5: The component-level reduction rule 5 is timing property preserving.

Proof: Since $port1^\bullet = port2^\bullet$, *port*1 and *port*2 share the same output transitions, this implies that only when both *port*1 and *port*2 get tokens can some transition in $L(E) - E_C.T$ be enabled. Therefore, we can prove this theorem by using the same method as that used in Theorem 1. □

Reduction rule 5 can be extended to cases with more than two input ports and more than two output ports.

It should be noted that

1) The component-level reduction rules are purely developed based on the external observable input–output patterns of components. In other words, no matter how complex the internal structure of a component is, a reduction rule may be applied to reduce the component as long as the component matches the pattern of the rule.
2) A component may be analyzed by use of reachability analysis method [2], [1], [15]. This is a fundamental and most widely applied method for analyzing TPNs. If necessary and possible, we can use some individual transition level reduction rules given in [8] to reduce the component before reachability analysis. In case a component is very complicated, we can also use simulation or test to obtain the timing parameters required by its reduced net.

In the next section, we illustrate how to build the CTPN model of a command and control system and apply the reduction rules to simplify the analysis of the response time of the system to its external arriving messages.

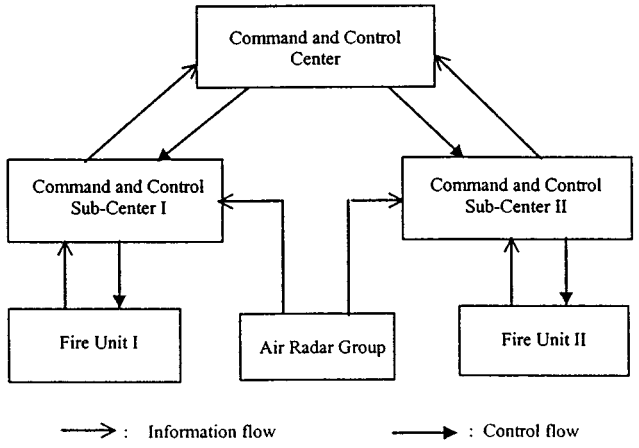


Fig. 7. Structure of a tactic anti-air C2 system.

IV. Modeling and Analysis of Command and Control Systems

A command and control (C2) system is a huge and complex integration of distributed hardware and software components. A C2 system is a typical real-time discrete event system. Excessive delays in execution of any of its functions may increase the damage probability, result in inefficient use of battle space, cause too many weapons to be assigned against the same target, and perhaps allow many targets to leak through a particular defense zone. This section illustrates the use of compositional TPNs to model and evaluate a tactic anti-air C2 system.

A typical structure of a tactic anti-air C2 system with two level command and control centers is shown in Fig. 7, which consists of a first-level command center (indicated as *C2 center*) and two second-level command centers (indicated as *subcenters*). An instance of (C2 center, subcenter) may be (division, regiment) or (brigade, battalion). They are geographically dispersed due to environmental and survivability reasons, leading to the distributed architecture of a C2 organization.

The operation of the system is described as follows.

S1) The radar group periodically sends a message containing air target information to the two subcenters at a rate that causes no queue at any stage of information processing and communication of the system.

S2) The C2 center is composed of three *seats*: two *intelligence seats* and one *decision-making seat*. The behavior specification of this component is as follows.

a) The two intelligence seats communicate with the decision-making seat through a common memory.

b) The messages from two subcenters are first copied and dispatched to the two intelligence seats. The two actions together take one to two time units.

c) The two intelligence seats then perform a situation assessment based on these messages to achieve a more precise situation picture, and then make a *threatening assessment* independently for each target and send the results to the decision-making seat. The first intelligence seat takes three to five time units to finish the two actions, and the second intelligence seat takes three to four time units.

d) The decision-making seat works on a scheme of *battle planning* (taking five to six time units). The result is sent to the two subcenters (taking one to two time units).

S3) The two subcenters have identical topology and timing properties. Each subcenter is composed of an intelligence seat and a decision-making seat. The behavior specification of this component is as follows.

a) The intelligence seat receives the message from its radar group and conducts *target discrimination*, *identification*, and *tracking*, and further conducts *threatening assessment*, then sends the result to the C2 center. It takes two to three time units.

b) After receiving the scheme of *battle planning* from the C2 center, the subcenter fuses it with related data in the database again (taking one to two time units) so as to form a detailed scheme of *weapon-to-target assignment* (taking five to seven time units). Furthermore, the results are sent to fire units (taking one to two time units).

S4) The two fire units have identical topology and timing properties. Their specification is as follows.

a) When the scheme of *weapon-to-target assignment* arrives from its subcenter, the unit first conducts *engagement control*. To this end, two computers concurrently compute shoot parameters (each computer taking two to four time units), and a third computer is responsible to fuse these parameters to form a complete engagement control command (taking two to three time units).

b) Then, it conducts *damage assessment* (taking five to seven time units), and feeds back the assessment results to its corresponding subcenter in time (taking one to two time units).

In this case study, we focus on timing requirements on the system. These requirements include:

R1) The system reaction time, i.e., the time delay from an enemy intelligence message being received by any subcenter to a fire command agaisnt the enemy being issued by a corresponding fire unit, must be less than or equal to 45 time units.

R2) Since the bottleneck for information processing is often located in component C2 center, the component is always asked to respond as soon as possible. This is captured by the requirement that the whole processing time for a group of messages from the two subcenters must be less than or equal to 22 time units.

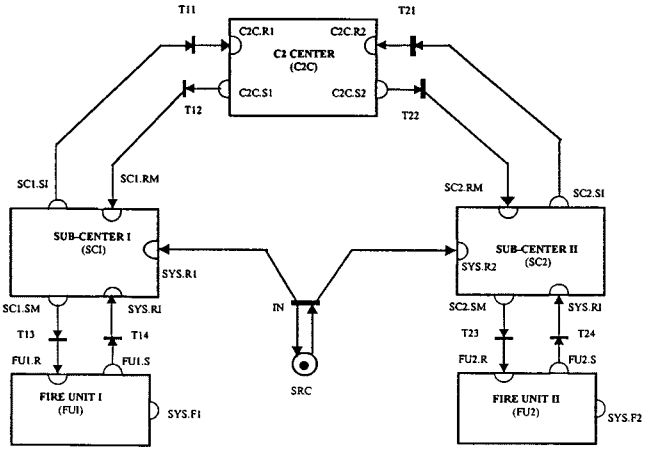


Fig. 8. Architecture of the CTPN model of the C2 system.

TABLE I
LEGENDS OF PARTIAL PORTS IN FIG. 8

Port	Type	Description
SYS.R1	Input	A message from Air Radar Group I arrived
SYS.F1	Output	A combat command to Fire Unit I sent
SC1.SI	Output	Sub-Center I ready to send intelligence to Command and Control Center
SC1.RM	Input	Sub-Center I receives command from Command and Control Center
SC1.SM	Output	Sub-Center I ready to send command to Fire Unit I
SC1.RI	Input	Sub-Center I receives damage assessment result from Fire Unit I
FU1.R	Input	Fire Unit I receives command from Sub-Center I
FU1.S	Output	Fire Unit I ready to send damage assessment result to Sub-Center I
C2C.R1	Input	Command and Control Center receives message from Sub-Center I
C2C.S1	Output	Command and Control Center ready to send command to Sub-Center I

A. Compositional TPN Model of the System

Fig. 8 shows the architecture of the CTPN model of the C2 system. The model has five components in total: a C2 center (C2C), two subcenters (SC1, SC2), and two fire units (FU1, FU2). Air-radar group is modeled as part of environment. Tables I and II show the legends of all nodes. Because of the similarity, we only describe the ports and transitions for C2 center, subcenter I, fire unit I and connections among them. Also assume that the two subcenters always obtain external inputs at the same time.

Now, we turn to the internal representation of each component. First let us consider the C2 center. Its TPN model is shown in Fig. 9. The messages from two subcenters are first copied and dispatched to the two intelligence seats (t101 fires). The two intelligence seats are responsible for performing *information fusion* and *threatening assessment* (t102 and t103 fire). The decision-making seat works on a scheme of *battle planning* (t104 fires). The result through ports C2C.S1 and C2C.S2 is sent to the two subcenters (T12 and T22 in two connectors fires). Note that the required synchronization among messages from the two subcenters has been modeled by transition t101, which does not fire until messages from both centers have been received.

TABLE II
LEGENDS OF NODES IN CONNECTORS IN FIG. 8

Nodes	Description	Firing Time
T11	Sub-Center I sends information to Command and Control Center	[1, 1]
T12	Command and Control center sends command to Sub-Center I	[1, 1]
T13	Sub-Center I sends command to Fire Unit I	[1, 1]
T14	Fire Unit I sends the result of loss assessment to Sub-Center I	[1, 1]
SRC	Radar group ready to send data to sub-centers	- -
IN	Radar group sends data to sub-centers	- -

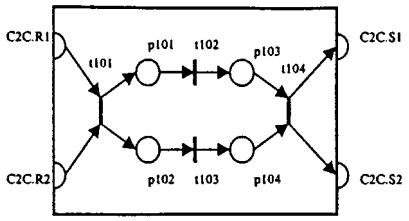


Fig. 9. TPN model of component C2 center. Firing times: t101: $[1, 2]$, t102: $[3, 5]$, t103: $[3, 4]$, t104: $[5, 6]$.

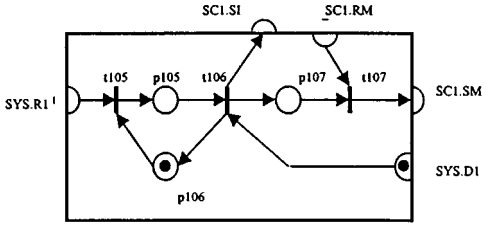


Fig. 10. TPN model of component subcenter I. Firing times: t105: $[2, 3]$, t106: $[1, 2]$, t107: $[4, 6]$.

Fig. 10 shows the TPN model of component subcenter I. The intelligence seat receives the message from its radar group and conducts *target discrimination*, *identification*, and *tracking* (t105 fires), and further conducts *threatening assessment* (t106 fires), then sends the result to the C2 center (T11 in a connector fires). After receiving the scheme of *battle planning* from C2 center (SC1.RM is marked), the subcenter fuses it with related data in the database again so as to form a detailed scheme of *weapon-to-target assignment* (t107 fires). Furthermore, the results are sent to fire units (T13 in Fig. 8 fires).

As shown in Fig. 11, the TPN model of fire unit 1 is composed of four transitions, five regular places and three ports. The information of weapon-to-target assignment from subcenter I is dispatched to the two computers (t108 fires), which concurrently compute shooting parameters (t109 and t110 fire). Then, the third computer performs result fusion (t111 fires). Then, the unit conducts *damage assessment* (t112 fires), and feeds back the assessment results to its corresponding subcenter in time (T14 in Fig. 8 fires).

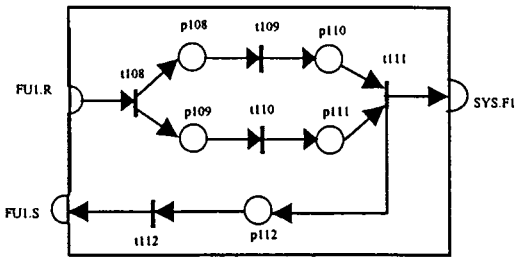


Fig. 11. TPN model of component fire unit I. Firing times: t108: $[1, 1]$, t109: $[2, 4]$, t110: $[2, 4]$, t111: $[2, 3]$, t112: $[5, 7]$.

B. Analysis of the C2 System

Applying the reduction rules involves timeliness analysis of TPNs. In [15], we present a new effective reachability analysis technique for timeliness of TPN models. The technique is based on a concept called *clock-stamped state class* (CS-class), which not only groups system states into compact representation of state classes but also records the time, relative to the beginning of system execution, when such states are reached. In particular, a CS-class consists of three parts: 1) a marking that represents a logical state of the modeled system; 2) a "global" firing domain corresponding to firing intervals, whose values are counted relative to the beginning of the net's execution, of all *firable* transitions in the state class; and 3) a clock stamp that corresponds to the moment when the state class is reached with the clock value relative to the beginning of the execution. We developed an algorithm to construct the reachability tree of the TPN based on the CS-class concept. With the reachability tree generated based on CS-classes, we can straightforwardly compute the time span between any two reachable CS-classes, thus the end-to-end time delay in task execution.

It follows from specification (S1) that the system takes the same statistical property of time to process inputs arriving at different times. This enables us to consider only one set of inputs, which are generated by firing transition *IN*, to verify the system requirements. In this case, for any component to be reduced, no transitions in it will be enabled by the coming tokens at its input ports until the previous input tokens have reached its output ports, which makes our reduction rules applicable. Applying component-level reduction rules 5 and 4 to component C2 center and fire unit I results in Fig. 12(a) and (b), respectively. Applying the enumerative analysis method [15], we conclude

$$SI(T\_\mathrm{C2C}) = [16, 20]$$
$$SI(T\_\mathrm{FU11}) = [5, 8], \quad \text{and}$$
$$SI(T\_\mathrm{FU12}) = [10, 15].$$

Thus, the requirement (R2) is verified.

Now, the reduced TPN model of the system to check requirement (R1) is shown in Fig. 13. Our goal is to compute the time delay that TPN runs from the initial state to the first marking in which places SYS.F1 and SYS.F2 are marked. Based on Fig. 13, reachability analysis shows that it takes 31 to 42 time units for the system to reach that state in which SYS.F1 and SYS.F2 are marked. This implies that requirement (R1) is satisfied.

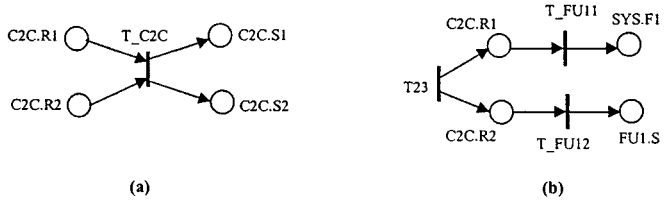


Fig. 12. (a) Reduction of component C2 center. (b) Reduction of component fire unit I.

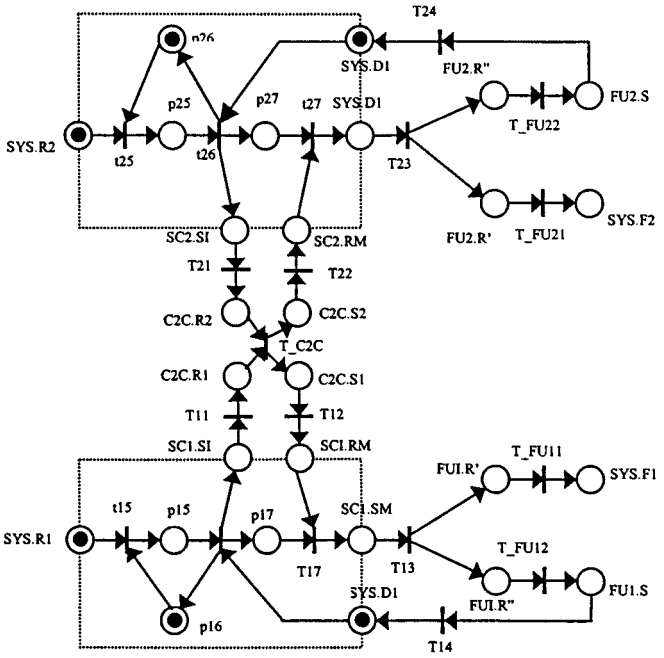


Fig. 13. TPN model for verifying requirement (R1).

Notice that in this example, we first reduced three components (C2C, FU1, and FU2) into three simple TPNs, so as to use a much simpler but equivalent model to analyze the system's timing property. The efficiency of the reduction depends on the complexity of the net inside the components to be reduced. In other words, the more complex a component, the more important the reduction.

Note that the reduction rules [8] work on individual transitions or places, and may not be applicable in some cases. For example, no such reduction rule is applicable for any reduction effort to component C2C.

V. Concluding Remarks

Compositional time Petri nets (CTPNs) augment the basic Petri net model with timing constraints of time Petri nets (TPNs) and a modular construct. CTPNs allow the decomposition of a complex model into several simple submodels, or components, and thus ease the modeling of complex real-time systems. To conquer the analysis complexity of TPNs, this paper also presented a set of component-level reduction rules for TPNs. Each of these reduction rules transforms a TPN component to a simple TPN while maintaining the net's external observable timing properties. To automate the analysis of CTPNs, we implemented a software tool named CTPNa&v. The tool is written

in C++ code and developed with UIM/X (professional edition), and presently runs on Solaris.

Introduction of time into a transition or place increases both the modeling power and the complexity of the net analysis. As a consequence, the reduction rules presented in this paper are only applicable to safe CTPNs. However, we believe that CTPN models and reduction rules are still helpful in timing property verification of complex real-time systems, as shown in Section IV. For an unsafe bounded CTPN, we can transform it to a safe CTPN by building the submodels of queues for possible buffers [17]. The cost for this transformation is the increase of the TPN size of component models. Currently, we are investigating reduction rules for CTPN's with multiple arriving tokens to components when the worst case is of interest in verifying real-time systems. For example, if a CTPN model is known k-bounded, we may design a worst external input scenario to any given component, and then analyze and reduce the component.

APPENDIX
PROOF OF THEOREM 1

Let time Petri net E' be derived from time Petri net E by applying component-level reduction rule 1. We need to prove both $L(E)-E_C.T \subseteq L(E')-\{t\}$ and $L(E')-\{t\} \subseteq L(E)-E_C.T$. We first prove $L(E)-N_C.T \subseteq L(E')-\{t\}$. We will show that if $\omega' \in L(E)-E_C.T$ then $\omega' \in L(E')-\{t\}$. Because any such ω' can be expressed as $\omega - E_C.T$ where $\omega \in L(E)$, it suffices to show that for any $\omega \in L(E), \omega' \in L(E')-\{t\}$ where $\omega' = \omega - E_C.T$.

We break our proof into several cases.

Case 1) *Port*1 doesn't get marked during ω.

We have assumed that the component has no enabled transition under the initial marking of E. Thus, in this case, we can conclude that none of the transitions in the component gets enabled during ω. Therefore, $\omega \in L(E)-E_C.T$, which implies that $\omega' \in L(E')-\{t\}$ because $E-E_C = E'-\{port1, port2, t\}$.

Case 2) *port*1 gets marked but *port*2 doesn't get marked during ω.

Let ω_1 be the shortest prefix of ω such that *port*1 gets marked during ω_1, which implies that $\omega_1 - E_C.T = \omega_1$, then we can write $\omega = \omega_1\omega_2$. Then each state in firing schedule ω_2 is reached by the independent firings of transitions in set $E_C.T$ and transitions in $E.T-E_C.T$, i.e., any transition that appears in ω_2 and belongs to $E_C.T$ is concurrent with those that appear in ω_2 but do *not* belong to $E_C.T$. It results in both $\omega_1(\omega_2 - E_C.T) \in L(E)$ and $\omega_1(\omega_2 - E_C.T) \in L(E')-\{t\}$. Since we have assumed that $\omega_1 - E_C.T = \omega_1$, so $\omega_1(\omega_2 - E_C.T) = \omega'$ because $E-E_C = E'-\{port1, port2, t\}$. Therefore, $\omega' \in L(E')-\{t\}$.

Case 3) Both *port*1 and *port*2 get marked once and only once during ω.

Let ω_1 be the shortest prefix of ω such that *port*1 gets marked during ω_1, which implies that $\omega_1 - E_C.T = \omega_1$, and $\omega_1\omega_{21}$ be the shortest prefix of such that *port*2 gets marked during ω. Then we can write $\omega = \omega_1\omega_{21}\omega_{22}$, where $\omega_{22} - E_C.T = \omega_{22}$ since we have assumed that *port*1 gets marked once and only once during ω. Let θ_1 be the time of $\varphi(E,\omega_1)$ and θ_{21} be the time of $\varphi(E,\omega_1\omega_{21})$. Then, from preconditions 1) and 2) of this rule, $\theta_{21}-\theta_1$ represents the time delay of the message transfer between *port*1 and *port*2, i.e., $\theta_{21}-\theta_1 \in SI\_(port1, port2), \max(\theta_{21}-\theta_1) = \sup(SI\_(port1, port2))$, and $\min(\theta_{21}-\theta_1) = \inf(SI\_(port1, port2))$. Since $SI(t) = SI\_(port1, port2), \theta_{21}-\theta_1 \in SI(t), \max(\theta_{21}-\theta_1) = \sup(SI(t)$, and $\min(\theta_{21}-\theta_1) = \inf(SI(t)$. Also, it follows from $\omega_1 - E_C.T = \omega_1$ and $E-E_C = E'-\{port1, port2, t\}$ that $\omega_1 \in L(E')$ and t is enabled under state $\varphi(E',\omega_1)$. Hence, we have $\omega_1(\omega_{21} - E_C.T)(t,\theta_{21}) \in L(E')$, and again since $E-E_C = E'-\{port1, port2, t\}$, the state $\varphi(E',\omega_1(\omega_{21} - E_C.T)(t,\theta_{21})$ is followed by ω_{22}. It follows from $\omega_{22} - E_C.T = \omega_{22}$ that $\omega_1(\omega_{21} - E_C.T)(t,\theta_{21})\omega_{22} \in L(E'_C)$, or $\omega_1(\omega_{21} - E_C.T)\omega_{22} \in L(E'_C) - \{t\}$, or $\omega' \in L(E')-\{t\}$.

Case 4) General case.

Suppose that *port*1 gets marked a times and *port*2 gets marked b times during ω. It follows from precondition (2) of this rule that either $a = b$ or $a = b+1$. We have proven the cases in which $a = 0, (a = 1)\wedge(b = 0)$, and $(a = 1)\wedge(b = 1)$. By the recursive use of the proofs for Case 2 and Case 3, we can prove that $\omega' \in L(E')-\{t\}$ for any schedule ω in which b takes any positive integral value.

Now we prove $L(E')-\{t\} \subseteq L(E)-E_C.T$. Similarly, we will show that for any $\omega \in L(E')$ holds $\omega' \in L(E)-E_C.T$ where $\omega' = \omega - \{t\}$.

We also break our proof in several cases.

Case 1) *Port*1 does not get marked during ω.

Because only when *port*1 gets marked can t fire, so in this case, we can conclude that t never fires during ω. Therefore, $\omega \in L(E')-\{t\}$, which implies that $\omega' \in L(E)-E_C$ because $E-E_C = E'-\{port1, port2, t\}$.

Case 2) *Port*1 gets marked but *port*2 doesn't during ω.

The assumption that *port*2 doesn't get marked implies that t never fires during ω. So, as true in Case 1, we also have $\omega' \in L(E)-E_C$.

Case 3) *Port*2 gets marked once and only once during ω.

Let ω_1 be the shortest prefix of ω such that *port*1 gets marked during ω_1, which implies that $\omega_1 - \{t\} = \omega_1$, and $\omega_1\omega_{21}$ be the shortest prefix of ω such that *port*2 gets marked during ω. Then, we can write $\omega = \omega_1\omega_{21}\omega_{22}$, where $\omega_{22} - \{t\} = \omega_{22}$ since we have assumed that *port*2 gets marked only once during ω, which implies that t only fires once. Let θ_1 be the time of $\varphi(E',\omega_1)$ and θ_{21} the time of $\varphi(E',\omega_1\omega_{21})$. Then, from the structure of the reduced net, we know

that $\theta_{21} - \theta_1$ represents the firing time interval of t, i.e., $\theta_{21} - \theta_1 \in SI(t), \max(\theta_{21} - \theta_1) = \sup(SI(t))$, and $\min(\theta_{21} - \theta_1) = \inf(SI(t))$. Since $SI(t) = SI\_(port1, port2)$, we have $\theta_{21} - \theta_1 \in SI\_(port1, port2), \max(\theta_{21} - \theta_1) = \sup(SI\_(port1, port2))$, and $\min(\theta_{21} - \theta_1) = \inf(SI\_(port1, port2))$. Also, it follows from $\omega_1 - \{t\} = \omega_1$ and $E - E_C = E' - \{port1, port2, t\}$ that $\omega_1 \in L(E)$ and $\varphi(E, \omega_1)$ enables at least one of the transitions in set $E_C.T$. Hence, we have $\omega_1(\omega_{21} - \{t\})\omega_c \in L(E)$, where ω_c is any firing schedule (segment) in E starting from *port*1 being marked and ending in *port*2 being marked and with only transitions in $E_C.T$ included. Again, since $E - E_C = E' - \{port1, port2, t\}$, the state $\varphi(E, \omega_1(\omega_{21} - \{t\})\omega_c)$ is followed by ω_{22}. It follows from $\omega_{22} - \{t\} = \omega_{22}$ that $\omega_1(\omega_{21} - \{t\})\omega_c\omega_{22}L(E_C)$, or $\omega_1(\omega_{21} - \{t\})\omega_c\omega_{22} \in L(E_C) - N_C.T$, or $\omega' \in L(E) - E_C.T$.

Case 4) General case.

Suppose that *port*1 gets marked a times and *port*2 gets marked b times during ω. It follows from the structure of the reduced net E' that either $a = b$ or $a = b + 1$. We have proved the cases in which $a = 0, (a = 1) \wedge (b = 0)$, and $b = 1$. By the recursive use of the proofs for Case 2 and Case 3, we can prove that $\omega' \in L(E) - E_C$ for any schedule ω in which b takes any positive integral value. □

ACKNOWLEDGMENT

The authors thank the anonymous reviewers for their comments and suggestions.

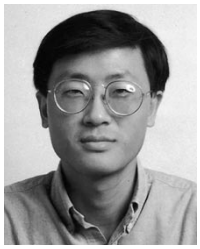
REFERENCES

[1] B. Berthomieu and D. Diaz, "Modeling and verification of time dependent systems using time Petri nets," *IEEE Trans. Softw. Eng.*, vol. 17, no. 3, pp. 259–275, 1991.
[2] B. Berthomieu and M. Menasche, "An enumerative approach for analyzing time Petri nets," in *Proc. IFIP Congr.*, Paris, France, Sept. 1983.
[3] G. Bucci and E. Vivario, "Compositional validation of time-critical systems using communicating time Petri nets," *IEEE Trans. Softw. Eng.*, vol. 21, no. 12, pp. 969–992, 1995.
[4] G. Florin, C. Fraize, and S. Natkin, "Stochastic Petri nets: Properties, applications and tools," *Microelectron. Reliab.*, vol. 31, no. 4, pp. 669–697, 1991.
[5] P. Merlin and D. Farber, "Recoverability of communication protocols—Implication of a theoretical study," *IEEE Trans. Commun.*, vol. COM-24, pp. 1036–1043, Sept. 1976.
[6] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, 1989.
[7] W. Reisig, *Petri Nets*. Berlin, Germany: Springer-Verlag, 1985.
[8] R. Sloan and U. Buy, "Reduction rules for time Petri nets," *Acta Inform.*, vol. 33, pp. 687–706, 1996.
[9] I. Suzuki and T. Murata, "A method for step wise refinements and abstractions of Petri nets," *J. Comput. Syst. Sci.*, vol. 27, no. 1, pp. 51–76, 1983.
[10] J. Tsai, S. Yang, and Y. Chang, "Timing constraint Petri nets and their application to schedulability analysis of real-time system specifications," *IEEE Trans. Softw. Eng.*, vol. 21, no. 1, pp. 32–49, 1995.
[11] R. Valette, "Analysis of Petri nets by stepwise refinements," *J. Comput. Syst. Sci.*, vol. 18, pp. 35–46, 1979.
[12] J. Wang, *Timed Petri Nets: Theory and Application*. Norwell, MA: Kluwer, 1998.
[13] J. Wang and Y. Deng, "Incremental modeling and verification of flexible manufacturing systems," *J. Intell. Manuf.*, vol. 10, no. 6, pp. 845–502, 1999.
[14] J. Wang, X. He, and Y. Deng, "Introducing software architecture specification and analysis in SAM through an example," *Inform. Softw. Technol.*, vol. 41, no. 7, pp. 451–567, 1999.
[15] J. Wang, G. Xu, and Y. Deng, "Reachability analysis of real-time systems using time Petri nets," *IEEE Trans. Syst., Man, Cybern. B*, 2000, to be published.
[16] M. C. Zhou and F. DiCesare, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*. Norwell, MA: Kluwer, 1993.
[17] ——, "Petri net modeling of buffers in automated manufacturing systems," *IEEE Trans. Syst., Man, Cybern. B*, vol. 26, no. 1, pp. 157–164, 1996.
[18] M. C. Zhou, *Petri Nets in Flexible and Agile Automation*. Norwell, MA: Kluwer, 1995.
[19] W. M. Zuberek, "Timed Petri nets: Definitions, properties, and applications," *Microelectron. Reliab.*, vol. 31, no. 4, pp. 627–644, 1991.
[20] R. Zurawski and M. C. Zhou, "Petri nets and industrial applications: A tutorial," *IEEE Trans. Ind. Electron.*, vol. 41, pp. 567–583, Dec. 1994.

Jiacun Wang received the M.S. and Ph.D. degrees in electrical and computer engineering from Nanjing University of Science and Technology (NUST), China, in 1989 and 1991, respectively.

He is currently Research Associate in the School of Computer Science, Florida International University (FIU), Miami. He was previously Associate Professor at NUST. His main research interests include software engineering, discrete event systems, formal methods, and distributed information systems. He authored *Timed Petri Nets: Theory and Application* (Norwell, MA: Kluwer, 1998) and published more than 40 research papers in journals and conferences.

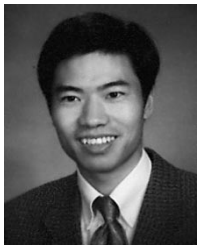
Dr. Wang served on the Program Committees of the 1994 International Conference on Electronics and Information Technology, Beijing, China, 1997 IEEE International Conference on Systems, Man and Cybernetics, Orlando, FL, and 1998 IEEE International Conference on Systems, Man, and Cybernetics, San Diego, CA.



Yi Deng (M'99) received the Ph.D. degree in computer science from the University of Pittsburgh, Pittsburgh, PA, in 1992.

He is Associate Professor of computer science and Managing Director of the Embedded Software Center (ESC), a joint university-industry R&D consortium, at the University of Texas, Dallas. He was previously Associate Professor and founding Director of the Center for Advanced Distributed System Engineering (CADSE) at Florida International University, Miami, a university-wide research center located in the School of Computer Science. His research interests include software architecture, distributed object technology, secure enterprise systems, and formal methods for complex software systems. He has published more than 45 papers in various journals and refereed conferences in the above areas. He is Associate Editor for the *International Journal of Software Engineering and Knowledge Engineering*.

Dr. Deng is a Referee and Program Committee Member for several journals and conferences. He is Program Committee Co-Chair for the 10th International Conference on Software Engineering and Knowledge Engineering. He is a Member of ACM.



MengChu Zhou (S'88–M'90–SM'93) received the B.S. degree in computer and control engineering from Nanjing University of Science and Technology, Nanjing, China, in 1983, the M.S. degree in automatic control from Beijing Institute of Technology, Beijing, China, in 1986, and the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1990.

He joined the Department of Electrical and Computer Engineering, New Jersey Institute of Technology (NJIT), Newark, in 1990, where he is currently Professor and Director of Discrete Event Systems Laboratory. He was Assistant Engineer in the Institute for Computer Applications, Beijing, from 1986 to 1987. His research interests include computer-integrated manufacturing, embedded control, discrete event systems, Petri nets and applications, life-cycle engineering, and intelligent automation. He co-authored, with F. DiCesare, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems* in 1993 (Norwell, MA: Kluwer, 1993), edited *Petri Nets in Flexible and Agile Automation* (Norwell, MA: Kluwer, 1995), and co-authored, with K. Venkatesh, *Modeling, Simulation, and Control of Flexible Manufacturing Systems: A Petri Net Approach* (Singapore: World Scientific, 1998). In addition, he has published more than 150 journal articles, book chapters, and conference proceeding papers in his research areas. He has organized and chaired more than 40 technical sessions and tutorials/workshops and served on program committees for many international conferences.

Dr. Zhou is an elected AdCom member of IEEE Systems, Man and Cybernetics Society, and Immediate Past President of Chinese Association for Science and Technology-USA. He was the recipient of the National Science Foundation's Research Initiation Award, and was listed in 1994 Computer Integrated Manufacturing University-LEAD Award by Society of Manufacturing Engineers and 2000 *Marquis Who's Who in Science and Engineering*, 5th Edition. He was granted the H. J. Perlis Research Award by NJIT in 1996. He served as Program Chair of *the 9th International Conference on CAD/CAM, Robotics, and Factories of the Future*, Newark, NJ, August 1993, 1997, *IEEE International Conference on Emerging Technologies and Factory Automation*, Los Angeles, CA, September 1997, and *1998 IEEE International Conference on Systems, Man, and Cybernetics*, San Diego, CA, October 1998. He is founding Chair of the Discrete Event Systems Technical Committee, IEEE Systems, Man, and Cybernetics Society. He served as Guest Co-Editor for IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS Special Section on Petri Nets in Manufacturing, and IEEE TRANSACTIONS ON SEMICONDUCTOR MANUFACTURING Special Section on Petri Nets in Semiconductor Manufacturing; *Journal of Intelligent Manufacturing* Special Issue on Computer Integrated Manufacturing Systems, *International Journal of Intelligent Control and Systems* Special Issue on Modeling and Control of Flexible Manufacturing Systems Using Petri Nets, and IEEE TRANSACTIONS ON SYSTEMS, MAN and CYBERNETICS Special Issue on Discrete Systems and Control. He is on the Editorial Board of IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION and *International Journal of Intelligent Control and Systems*.