Compositional Schedulability Analysis of Real-Time Systems Using Time Petri Nets

Dianxiang Xu, *Member*, *IEEE Computer Society*, Xudong He, *Member*, *IEEE Computer Society*, and Yi Deng, *Member*, *IEEE*

Abstract—This paper presents an approach to the schedulability analysis of real-time systems modeled in time Petri nets by separating timing properties from other behavioral properties. The analysis of behavioral properties is conducted based on the reachability graph of the underlying Petri net, whereas timing constraints are checked in terms of absolute and relative firing domains. If a specific task execution is schedulable, we calculate the time span of the task execution, and pinpoint nonschedulable transitions to help adjust timing constraints. A technique for compositional timing analysis is also proposed to deal with complex task sequences, which not only improves efficiency but also facilitates the discussion of the reachability issue with regard to schedulability. We have identified a class of well-structured time Petri nets such that their reachability can be easily analyzed.

Index Terms—Real-time systems, time Petri nets, schedulability, reachability.

1 Introduction

In a real-time system, the process of verifying whether a schedule of task execution meets the imposed timing constraints is referred to as schedulability analysis [10]. Many researchers have tackled this problem by focusing on either the implementation of a real-time system or the specification of a real-time system. Examples of schedulability analysis based on implementations include the works by Stoyenko et al. [10] and Haban and Shin [5]. In [10], a set of language-independent schedulability techniques based on the information of program implementation was proposed. In [5], an approach to monitor and verify the task executions was presented. Representatives of schedulability analysis based on specifications include the real time logic technique by Jahanian and Mok [6], and the Petri net based technique by Tsai et al. [11].

Our work studies schedulability analysis of specifications modeled in time Petri nets [8]. As a visual model, time Petri nets (TPNs) have been proven very convenient for expressing timing constraints in time dependent systems. TPNs associate transitions with time pairs instead of single delays in timed Petri nets, thus TPNs are more general than timed Petri nets [1]. Furthermore, TPNs support formal analysis by adapting the well-known reachability analysis technique [1], [2]. A reachability graph (or tree) provides a representation of the complete dynamic behavior of a TPN based on the interleaving semantics. The nodes are state classes and the edges are labeled with firing transitions and firing domains reflecting timing constraints. Schedulability, though closely related to reachability, has more specific concerns about transition sequences rather than markings or states. State classes constructed for the purpose of validating the dynamic behavior is therefore not so effective for schedulability analysis. In particular, the end-to-end delay in task execution, an important issue in time critical systems, cannot be directly derived from the firing domain of state classes. Thus, the techniques developed in [1], [2] are useful for reachability analysis but not efficient for schedulability analysis.

An alternative analysis technique for real-time systems is to separate the analysis of timing properties from the analysis of other nontiming behavioral properties. For a Petri net (PN) based model with extended time-handling capability, the analysis can be conducted in two phases: reachability analysis without considering the timing constraints and timing analysis of task sequences. Reachability is analyzed to verify whether a transition sequence δ is an occurrence sequence reaching a certain marking M_n in the underlying PN. The occurrence sequence δ is then analyzed to verify whether δ is schedulable or M_n is reachable by means of δ with the timing constraints. Tsai et al. employed this approach for the schedulability analysis of real-time system specifications modeled by timing constraint Petri nets (TCPNs) [11]. TCPNs extend Petri nets by associating a minimum/maximum timing constraint with each transition and place, and associating a duration constraint for firing each transition. Different from TPNs and timed PNs, TCPNs use the weak firing rule. TCPNs are more expressive, but more complicated to use. Furthermore, it is difficult to address the general reachability issue of TCPNs when we need to analyze both behavioral and timing properties. The schedulability analysis of TCPNs, though adapted from TPNs and timed PNs, is not applicable to that of TPNs because of different firing rules that have different interpretations of timing constraints on net structures such as synchronization and concurrence. In addition, the following formulas of earliest beginning fire time (EFBT) and latest fire ending time (LFET) for a weakly firable transition (WFT) in Definition 4 [11] are inconsistent with the meanings of timing constraints:

- *D. Xu is with the Department of Computer Science, Harvey R. Bright Building, Rm. 325, Texas A & M University, College Station, TX 77843-3112. E-mail: xudian@cs.tamu.edu.*
- *X. He and Y. Deng are with the School of Computer Science, Florida International University, Miami, FL 33199. E-mail: {hex, deng}@cs.fiu.edu.*

Manuscript received 3 Nov. 2000; revised 24 July 2001; accepted 15 Feb. 2002. Recommended for acceptance by B. Sanders.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number 113112.

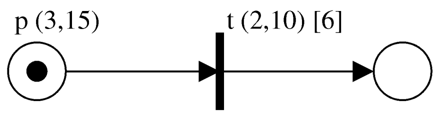


Fig. 1. TCPN—Example 1.

$$\mathrm{EFBT}(t_j) = \mathrm{Max}\{TC_{min}(p_j)\} + TC_{min}(t_j),$$
$$\mathrm{LFET}(t_j) = \mathrm{Min}\{TC_{max}(p_j), TC_{max}(t_j)\}.$$

This problem leads to the incorrect conclusions in Theorems 1 and 2.

Consider a simple example in Fig. 1, where p is the only input place of transition t. Suppose the token in p arrives at time T_0. Then, transition t is enabled at T_0.

Using the above formulas, $\mathrm{EFBT}(t) = 3 + 2 = 5$ and $\mathrm{LFET}(t) = \mathrm{Min}\{15, 10\} = 10$.

$$\mathrm{LFET}(t) - \mathrm{EFBT}(t) = 10 - 5 = 5 < \mathrm{FIRE}_{dur}(t) = 6.$$

Thus, t is not firable. As a matter of fact, this is not consistent with the meanings of timing constraints. The timing constraints on place p, $(TC_{min}(p), TC_{max}(p)) = (3, 15)$, are the minimum/maximum elapsed time intervals between the token arrival time of p (T_0), and the beginning/ending firing times of p's output transition, i.e., t. In other words, t can fire only during $(T_0 + 3, T_0 + 15)$. The timing constraints on t, $(TC_{min}(p), TC_{max}(p)) = (2, 10)$ mean that t is firable only during $(T_0 + 2, T_0 + 10)$. Thus,

$$\mathrm{EFBT}(t) = \mathrm{Max}\{T_0 + 3, T_0 + 2\} = T_0 + 3,$$

and

$$\mathrm{LFET}(t) = \mathrm{Min}\{T_0 + 15, T_0 + 10\} = T_0 + 10,$$
$$\mathrm{LFET}(t) - \mathrm{EFBT}(t) = (T_0 + 10) - (T_0 + 3) = 7 > 6.$$

Thus, t is firable.

In more general cases when token arrival times are considered, formulas 4.a and 4.b in Theorem 1 defining EFBT and LFET for a strongly firable transition (SFT) are incorrect either. For the example in Fig. 2, suppose the token arrival times of p_1, p_2, and p_3 be different.

Considering the place constraints, t can only fire in the following three intervals:

$$(\mathrm{TOKEN}_{arr}(p_j) + TC_{min}(p_j), \mathrm{TOKEN}_{arr}(p_j) + TC_{max}(p_j)),$$

where $p_j \in \{p_1, p_2, p_3\}$. On the other hand,

$$\mathrm{MAX}\{\ \mathrm{TOKEN}_{arr}(p_j) : p_j \in \{p_1, p_2, p_3\}\}$$

is the time when t is enabled. According to the transition constraints, t is firable only during

$$(\mathrm{MAX}\{\ \mathrm{TOKEN}_{arr}(p_j) : p_j \in \{p_1, p_2, p_3\}\} + TC_{min}(t),$$
$$\mathrm{MAX}\{\ \mathrm{TOKEN}_{arr}(p_j) : p_j \in \{p_1, p_2, p_3\}\} + TC_{max}(t)).$$

Thus,

$$\mathrm{EFBT}(t) = \mathrm{MAX}\{\ \mathrm{TOKEN}_{arr}(p_j) + TC_{min}(p_j),$$
$$\mathrm{MAX}\{\ \mathrm{TOKEN}_{arr}(p_j) : p_j \in \{p_1, p_2, p_3\}\} + TC_{min}(t)\}$$
$$\mathrm{LFET}(t) = \mathrm{MIN}\{\ \mathrm{TOKEN}_{arr}(p_j) + TC_{max}(p_j),$$
$$\mathrm{MAX}\{\ \mathrm{TOKEN}_{arr}(p_j) : p_j \in \{p_1, p_2, p_3\}\} + TC_{max}(t)\}.$$

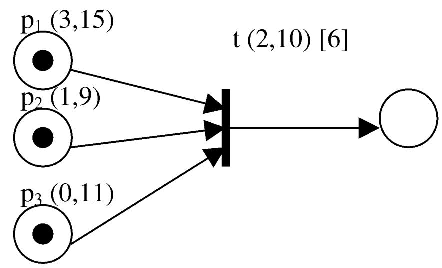


Fig. 2. TCPN—Example 2.

These should be the correct versions for formula 4.a and 4.b in Theorem 1. However, they are not as easy to deduce $\mathrm{TOKEN}_{arr}(p_j)$ as in formulas 4.a and 4.b. In other words, it is rather complex to use them to automatically determine the schedulability of all transitions in a TCPN. In conclusion, the complex timing constraints provide little help for modeling and analyzing real time systems.

In this paper, we focus on the schedulability analysis of TPNs. Our main results include: 1) an approach for determining whether a specific transition sequence is schedulable or not, for calculating the time span of a schedulable task execution, or for pinpointing out non-schedulable transitions to help adjust timing constraints and correct design errors, 2) a compositional approach to deal with complex task sequences, 3) identification of a class of well-structured time Petri nets so that the reachability of these nets can be easily analyzed. These results serve dual purposes: On one hand, TPNs are used as a model for architectural specification in SAM [13], a software architecture specification model developed by us, the scheduability technique provides an important analysis technique for timing critical properties of SAM specifications. On the other hand, our schedulability analysis technique of TPNs offers a more effective and practical way complementing the traditional reachability analysis. Our schedulability analysis is based on both relative and absolute time modes and can be integrated with reachability analysis of TPNs [1], [14].

This paper is organized as follows: Section 2 gives a brief introduction to TPNs and schedulability. Section 3 shows how to conduct timing verification for schedulability analysis of task execution by separating timing properties from behavioral properties. Section 4 describes how to conduct schedulability analysis by decomposing a firing sequence in underlying Petri net into a number of subsequences. Section 5 discusses the reachability problem of TPNs based on the reachability graph of underlying Petri net and the compositional schedulability analysis. Section 6 demonstrates the main idea through an example. Sections 7 concludes the paper.

2 Time Petri Nets and Schedulability

A time Petri net TN is a tuple (P, T, B, F, C, M_0) where:

- P is a finite set of places.
- T is a finite set of transitions.
- B is the backward incidence function, $B: T \times P \to N$, where N is the set of nonnegative integers.
- F is the foreward incidence function, $F: T \times P \to N$.
- M_0 is the initial marking function, $M_0: P \to N$.

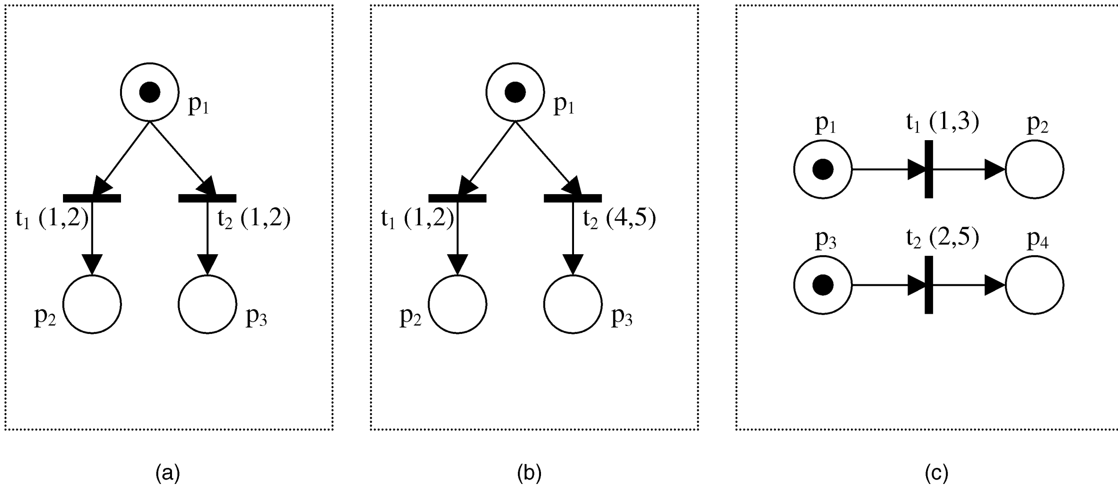


Fig. 3. Schedulability of transitions.

- C is a mapping called static interval, $C: T \rightarrow Q^* \times Q^*$ where Q^* is the set of nonnegative rational numbers.

P, T, B, F, and M_0 together define a Petri net without any timing constraints. We denote this underlying net as $UN = (P, T, B, F, M_0)$ and use its reachability graph as the basis for the schedulability analysis of TN. Given a marking M and a place $p \in P$, M(p) denotes the number of tokens in p. For any transition $t \in T$, ${}^\bullet t = \{p \in P\colon (t, p) \in B\}$. For convenience, we also denote C(t) as $t(EFT(t), LFT(t)) \in C$, where $t \in T$, EFT, and LFT are called static (relative) earliest firing time and latest firing time, respectively. The static interval for any transition is finite since this paper is mainly concerned with the schedulability of finite task sequences within finite time. Let $I_1 = (a_1, b_1)$, $I_2 = (a_2, b_2)$, where $0 \leq a_i \leq b_i < \infty$ and $k \in N$, we define $I_1 + I_2 = (a_1 + a_2, b_1 + b_2)$, $I_1 - I_2 = (a_1 - a_2, b_1 - b_2)$, and $k^* I_1 = (k^* a_1, k^* b_1)$.

In a time Petri net TN, a transition t is said to be *enabled* under marking M if $(\forall p \in {}^\bullet t) M(p) \geq B(t, p)$. An enabled transition t with time interval t(EFT (t), LFT (t)) under marking M at time τ may not fire before $\tau + EFT(t)$ and must fire before or at $\tau + LFT(t)$ unless another transition fires before and modifies M [1]. According to the *strong firing mode*, a transition is forced to fire at $\tau + LFT(t)$ if the transition has not fired and not been disabled by others transitions' firing (on the contrary, PNs use a *weak firing mode*, which does not force an enabled transition to fire; in other words, an enabled transition may or may not fire) [11]. As in [1], we also assume no transition can be multiply enabled. We use EN(M) to denote the set of transitions enabled under marking M.

In a TN, an enabled transition t is said to be *schedulable* under marking M if t can be the first transition to fire (i.e., can fire before any other enabled transitions). For example, in Fig. 3a, both t_1 and t_2 in the conflict structure are schedulable under the current marking, though the firing of one transition makes the other disabled under the new marking. In Fig. 3b, only t_1 is schedulable; t_2 is not schedulable because t_1 must fire before t_2 has a chance to fire. In Fig. 3c, both t_1 and t_2 are schedulable. After the firing of t_1 (or t_2), t_2 (or t_1) is still enabled and schedulable under the new marking. Generally, the schedulability of an individual transition depends on the time constraints of all enabled transitions under the current marking. Transition t is schedulable under the initial marking M_0 if $EFT(t) \leq \min\{LFT(t')\colon t' \text{ is enabled under } M_0\}$. More general cases are discussed in the next section.

In a TN, a transition sequence $\delta = (t_1 \dots t_i \dots t_n)$ is said to be *schedulable* or δ is a *schedule* if all transitions in δ are schedulable in the given order, that is, there exist markings $M_1, \dots, M_n$ such that $(M_0 t_1 M_1 \dots t_i M_i \dots t_n M_n)$ is a firing sequence in the underlying net and $t_i (1 \leq i \leq n)$ is schedulable under M_{i-1}. If at least one transition in δ is nonschedulable, then δ is nonschedulable. In a TN, a marking M_n is said to be reachable from M_0 if there exists a schedule δ that reaches M_n from M_0. In the underlying net UN, a marking M_n is said to be reachable if there is a firing sequence $(M_0 t_1 M_1 \dots t_i M_i \dots t_n M_n)$, or simply an occurrence sequence $\delta = (t_1 \dots t_i \dots t_n)$, that transforms M_0 to M_n. We use $L(M_0, M_n)$ to denote the set of all possible firing sequences from M_0 to M_n in a UN. If a transition sequence is not an occurrence sequence in the UN, it is not schedulable in a TN. However, an occurrence sequence in the UN is not necessarily a schedule in the TN and a marking that is reachable in the UN is not necessarily reachable in the TN. In this paper, we focus on checking whether occurrence sequences in a UN are schedulable in the TN.

3 Schedulability Analysis

In this section, we first show that the time span of a task sequence cannot be accurately evaluated by the relative time mode for checking the schedulability in a TN. A method integrating the absolute time mode with the relative time mode is then presented to conduct a schedulability analysis.

Generally, the *relative firing domains* for the state classes of TPNs in [1] can be used to determine the schedulablility of transitions. Let us first disregard the timing inequalities on pairs of enabled transitions, which were originally used for comparing state classes when generating state class graphs. Suppose D_i is a relative firing domain for enabled transitions at M_i. The dynamic firing interval of transition t in D_i is denoted as $D_i(t)$, $(REFT_i(t), RLFT_i(t))$, or $t(REFT_i(t), RLFT_i(t))$, where $REFT_i(t)$ and $RLFT_i(t)$ are referred to as the relative earliest firing time and the relative latest firing time, respectively. Let $D_0 = \{C(t) : t \in EN(M_0)\}$ and (RE_i, RL_i) be the relative schedulable interval for t_i. The schedulability of $t_{i+1} (0 \le i \le n-1)$ in firing sequence $(M_0 t_1 \ldots\ldots t_n M_n)$ can be checked by the following steps:

Step 1: If $t_{i+1} \in EN(M_i)$ and $REFT_i(t_{i+1}) \le MIN\{RLFT_i(t) : t \in EN(M_i)\}$, then t_{i+1} is schedulable at marking M_i during interval

$$(RE_{i+1}, RL_{i+1}) = (RLFT_i(t_{i+1}), MIN\{RLFT_i(t) : t \in EN(M_i)\}).$$

Else, t_{i+1} is nonschedulable at marking M_i;

Step 2: Build new relative firing domain:

- $D_{i+1} := \emptyset$.
- For any newly enabled transition t (disabled under M_i and enabled under M_{i+1}), add its static time interval C(t) into D_{i+1}.
- For any inherited transition t (enabled by both M_i and M_{i+1}, $t \ne t_{i+1}$), the interval of t to be added into D_{i+1} is $(MAX\{0, REFT_i(t) - RL_{i+1}\}, RLFT_i(t) - RE_{i+1})$.
- If t_{i+1} is enabled again under M_{i+1}, add its static interval into D_{i+1},

Let us check the schedulability of $t_1 t_2$ in Fig. 3c, where $M_0 = \{p_1, p_3\}$ and $D_0 = \{t_1(1,3), t_2(2,5)\}$.

1. Check $t_1 : t_1$ is schedulable during $(1, \min\{3,5\}) = (1,3)$; $//M_1 = \{p_2, p_3\}$

$$D_1 = \{t_2(\max\{2-3, 0\}, 5-1)\} = \{t_2(0,4)\}.$$

2. Check $t_2 : t_2$ is schedulable during (0,4); $//M_2 = \{p_2, p_4\}$ $D_2 = \emptyset$.

According to the above steps, t_1 fires at $\theta_1 (1 \le \theta_1 \le 3)$ relative to time τ (at M_0) and t_2 fires at $\theta_2 (0 \le \theta_2 \le 4)$ relative to $\tau + \theta_1$(at M_1). At M_2 after t_2 fires, the time should be $\tau + \theta_1 + \theta_2$. Is $(1,3) + (0,4) = (1,7)$ the interval for $\theta_1 + \theta_2$? The answer is negative. At least, this is not accurate. The correct time span for $t_1 t_2$ is (2,5) because t_2 is enabled under M_0 and it must fire at sometime between (2,5). Similarly, $t_2 t_1$ is a schedule in Fig. 3c. The relative schedulable interval for t_2 is $(2, \min\{3,5\}) = (2,3)$ and the relative schedulable interval for t_1 is $(\max\{1-3, 0\}, 3-2) = (0,1)$. $(2,3) + (0,1) = (2,4)$ is not the correct time span for $t_2 t_1$ either. Since t_1 must fire before or at 3, the correct time span for $t_2 t_1$ should be (2,3). As a matter of fact, we cannot accurately determine the time span of an occurrence sequence according to the relative firing domains. The problem arises from the timing constraints of concurrent transitions. For a transition t enabled at both M_i and M_{i+1}, relative interval $(MAX\{0, REFT_i(t) - RL_{i+1}\}, RLFT_i(t) - RE_{i+1})$ loses time information for calculating the time span. For example, the relative interval at M_i is not necessarily equal to the relative interval at M_{i+1} plus the schedulable interval during which t_{i+1} fires under M_i, that is, $(REFT_i(t), RLFT_i(t))$ is not necessarily equal to

$$(MAX\{0, REFT_i(t) - RL_{i+1}\}, RLFT_i(t) - RE_{i+1}) + (RL_{i+1}, RE_{i+1}).$$

The above problem may be solved by transforming timing inequalities for transition pairs in dynamic firing domains into a canonical form before their use. However, this approach has a polynomial complexity [2] and is less understandable and effective. In this paper, we introduce the *absolute firing domains* for enabled transitions and global time stamps for reached markings based on absolute intervals, which are relative to τ at M_0. After firing a transition, the reached marking is stamped and an absolute firing domain is constructed for newly enabled transitions and those transitions remain enabled. Suppose $TS_i = (AE_i, AL_i)$ is the time stamp at M_i, which means M_i is reached at sometime during (AE_i, AL_i) relative to τ at M_0(M_i cannot be reached before AE_i or after AL_i); $AD_i (1 \le i \le n)$ is the absolute firing domain for the transitions enabled under M_i. The interval of transition t in AD_i is denoted as $AD_i(t)$, $(AEFT_i(t), ALFT_i(t))$, or $t(AEFT_i(t), ALFT_i(t))$, where $AEFT_i(t)$ and $ALFT_i(t)$ are referred to as the absolute earliest firing time and the absolute latest firing time, respectively. $AD_0(t) = \{C(t) : t \in EN(M_0)\}$; let the time stamp at M_0 be $TS_0 = (0,0)$. Whether firing sequence $(M_0 t_1 \ldots t_i M_i \ldots t_n M_n)$ in a UN is schedulable or not in $TN = (P, T, B, F, C, M_0)$ can be determined by checking each transition $t_{i+1} (0 \le i \le n-1)$ as follows:

Step 1: If $t_{i+1} \notin EN(M_i)$, then, δ is not an occurrence sequence in UN and, thus, nonschedulable.

Step 2: If $REFT_i(t_{i+1}) \le MIN\{RLFT_i(t) : t \in EN(M_i)\}$, then t_{i+1} is schedulable at marking M_i; do Steps 3-5. Else, t_{i+1} is nonschedulable at marking M_i.

Step 3: Calculate the relative schedulable interval of t_{i+1}:

$$(RE_{i+1}, RL_{i+1}) = (REFT_i(t_{i+1}), MIN\{RLFT_i(t) : t \in EN(M_i)\}).$$

Calculate the absolute schedulable interval of t_{i+1}, i.e., time stamp:

$$TS_{i+1} = (AE_{i+1}, AL_{i+1}) = (AEFT_i(t_{i+1}), MIN\{ALFT_i(t) : t \in EN(M_i)\}).$$

Step 4: Build the new relative firing domain D_{i+1} from D_i:

- $D_{i+1} := \emptyset$.
- For any newly enabled transition t (disabled under M_i and enabled under M_{i+1},), add its static time interval C(t) into D_{i+1}, i.e.,

$$D_{i+1} := D_{i+1} \cup \{C(t) : t \notin EN(M_i) \wedge t \in EN(M_{i+1})\}.$$

- For any inherited transition t (enabled by both M_i and M_{i+1}) and $t \ne t_{i+1}$, the interval of t to be added into D_{i+1} is

$$(MAX\{0, REFT_i(t) - RL_{i+1}\}, RLFT_i(t) - RE_{i+1}),$$

i.e.,

$$D_{i+1} := D_{i+1} \cup \{(MAX\{0, REFT_i(t) - RL_{i+1}\}, RLFT_i(t) - RE_{i+1}) : t \in EN(M_i) \wedge t \in EN(M_{i+1}) \wedge t \ne t_{i+1}\}.$$

- If t_{i+1} is enabled by M_{i+1} (t_{i+1} has already fired under M_i, but is enabled again under M_{i+1}), add its static interval into D_{i+1}, i.e.,

$$D_{i+1} := D_{i+1} \cup \{C(t_{i+1}) : t_{i+1} \in EN(M_{i+1})\}.$$

Step 5: Build the new absolute firing domain AD_{i+1} from AD_i:

- $AD_{i+1} := \emptyset$.
- For any newly enabled transition t, add the sum of its static interval and current time stamp, i.e.,

$$AD_{i+1} := AD_{i+1} \cup \{C(t) + TS_{i+1} : t \notin EN(M_i) \wedge t \in EN(M_{i+1})\}.$$

- For any inherited transition t $(t \neq t_{i+1})$, the interval added to AD_{i+1} is

$$(MAX\{AEFT_i(t), AE_{i+1}\}, ALFT_i(t)),$$

i.e.,

$$AD_{i+1} := AD_{i+1} \cup \{(MAX\{AEFT_i(t), AE_{i+1}\}, ALFT_i(t)) : t \in EN(M_i) \wedge t \in EN(M_{i+1}) \wedge t \neq t_{i+1}\}.$$

- If t_{i+1} is enabled under M_{i+1}, add the sum of its static interval and current time stamp into AD_{i+1}, i.e.,

$$AD_{i+1} := AD_{i+1} \cup \{C(t_{i+1}) + TS_{i+1} : t_{i+1} \in EN(M_{i+1})\}.$$

In the above algorithm, $EN(M_{i+1}) (0 \leq i \leq n-1)$ are directly obtained from the reachability tree of the underlying Petri net UN. Relative firing domains D_i are used to determine the schedulability of individual transitions and absolute firing domains AD_i are used to calculate the times when transitions fire and new markings are reached. The computations of AD_i and D_i are independent from the preconditions of Steps 1 and 2. The time stamp TS_{i+1} is the absolute schedulable interval of t_{i+1} reaching M_{i+1}. If the sequence is schedulable, the time span of executing the sequence is TS_n. This can be illustrated by the following induction: 1) According to the definition of schedulability, t_1 is schedulable during $(EFT(t_1), MIN\{LFT(t): t \in EN(M_0)\}$. Since $(EFT(t), LFT(t)) = C(t) = (AEFT_0(t), ALFT_0(t))$ for any $t \in EN(M_0)$, the schedulable interval of t_1 is equal to

$$(AEFT_0(t_1), MIN\{ALFT_0(t): t \in EN(M_0)\}) = TS_1 = (AE_1, AL_1).$$

TS_1 is the interval (time stamp) for reaching M_1. In addition, AD_1 contains correct absolute intervals for all enabled transitions under M_1. For any newly enabled transition t, the absolute interval during which t can fire is the static interval (relative to M_1,) plus the absolute schedulable interval of t_1 (the interval during which M_1 is reached, i.e., TS_1). For any transition t $(t \neq t_1)$ enabled under both M_0 and M_1, t will never fire before AE_1 because t_1 fires after or at AE_1 and t must fire after t_1. Thus, $(MAX\{AEFT_1(t), AE_1\})$ is the earliest absolute time that t can fire. If t_1 is still enabled under M_1, the new absolute interval for t_1 in AD_1 is its static interval plus TS_1 (like a newly enabled transition). 2) Suppose TS_i is the time span during which t_i fires and AD_i contains correct absolute intervals for all enabled transitions under M_i. Obviously, t_{i+1} may not fire before $AEFT_i(t_{i+1})$. Also, t_{i+1} must fire before any other enabled transition is forced to fire, i.e., t_{i+1} must not fire after $MIN\{ALFT_i(t) : t \in EN(M_i)\}$. Thus, $(AEFT_i(t_{i+1}), MIN\{ALFT_i(t) : t \in EN(M_i)\})$ is the interval during which t_{i+1} fires or the time span of firing $t_1, t_2, \ldots t_{i+1}$. This interval is exactly TS_{i+1}. Similarly, it is easy to show that AD_{i+1} also contains correct absolute intervals for all enabled transitions under M_{i+1}.

Furthermore, we can easily know the time span between any two transitions or markings in a schedulable occurrence sequence. In fact, the time span of the subsequence $(t_i t_{i+1} \ldots t_j)(j > i > 0)$ or from t_i to t_j is $TS_j - TS_{i-1}$. This facilitates the composition of transition sequences because a sequence that does not begin with M_0 can also be analyzed.

For example, the schedulability of $\delta = (t_1 t_2 t_3 t_4 t_5)$ in Fig. 4a is checked as follows:

1. Initial time stamp $TS_0 = (0, 0)$; $//M_0 = \{p_1\}$;
   $D_0 = \{t_1(0, 5)\}$;
   Initial absolute firing domain $AD_0 = \{t_1(0, 5)\}$.
2. Check $t_1 : t_1$ is schedulable during dynamic relative interval (0,5);
   New time stamp $TS_1 = (0, 5)$; $//M_1 = \{p_2\}$;
   $D_1 = \{t_2(1, 4), t_6(5, 7)\}$;
   Add the intervals of newly enabled transitions t_2 and t_6 to new absolute domain AD_1:
   $AD_1(t_2) = C(t_2) + TS_1 = (1, 4) + (0, 5) = (1, 9)$;
   $AD_1(t_6) = C(t_6) + TS_1 = (5, 7) + (0, 5) = (5, 12)$.
3. Check $t_2 : REFT_1(t_2) = 1 \leq MIN\{RLFT_1(t_2) = 4, RLFT_1(t_6) = 7\} = 4$;
   t_2 is schedulable during (1,4); $//M_2 = \{p_3, p_4\}$;
   $TS_2 = (1, 9)$;
   $D_2 = \{t_3(1, 3), t_4(4, 5)\}$;
   Add the intervals of newly enabled transitions t_3 and t_4 to new absolute domain AD_2:
   $AD_2(t_3) = C(t_3) + TS_2 = (1, 3) + (1, 9) = (2, 12)$;
   $AD_2(t_4) = C(t_4) + TS_2 = (4, 5) + (1, 9) = (5, 14)$.
4. Check $t_3 : REFT_2(t_3) = 1 \leq MIN\{RLFT_2(t_3) = 3, RLFT_2(t4) = 5\} = 3$
   t_3 is schedulable during (1,3); $//M_3 = \{p_5, p_4\}$;
   $TS_3 = (2, 12)$;
   $D_3 = \{t_4(1, 4)\}$; $AD_3(t_4) = (5, 14)$.
5. Check $t_4 : t_4$ is schedulable during (1,4);
   $//M_4 = \{p_5, p_6\}$;
   $TS_4 = (5, 14)$;
   $D_4 = \{t_5(1, 5)\}$;
   $AD_4(t_5) = (1, 5) + (5, 14) = (6, 19)\}$.
6. Check $t_5 : t_5$ is schedulable during (1,5);
   $//M_n = M_5 = \{p_7\}$;
   $TS_5 = (6, 19)$;
   $D_5 = \emptyset$; $AD_5 = \emptyset$.

Thus, δ is schedulable and the time span of δ is $TS_5 = (6, 19)$. The time span of subsequence $(t_3 t_4)$ is $TS_4 - TS_2 = (5, 14) - (1, 9) = (4, 5)$, i.e., it takes four to five units of time to finish firing t_3 and then t_4. This complies with the interpretation of timing constraints imposed on the

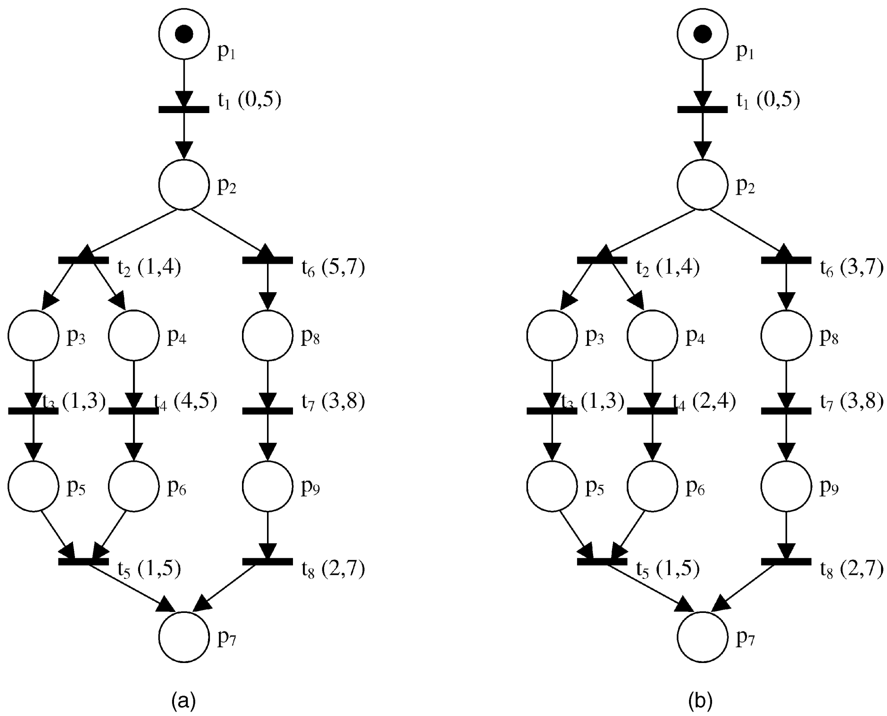


Fig. 4. Schedulability of transition sequences.

concurrence structure. Note that, the absolute firing domains cannot be used to determine the schedulability of individual transitions. In the above example, the absolute domain after firing t_1 is $AD_1 = \{t_2(1,9), t_6(5,12)\}$. It seems from AD_1 that either t_2 or t_6 in the synchronization/conflict structure is schedulable. This is not true because t_6 is nonschedulable according to the timing constraints defined for t_2 and t_6. Similarly, it seems from $AD_2 = \{t_3(2,12), t_4(5,14)\}$ that t_3 and t_4 in the concurrence structure can fire in both sequences: (t_3t_4) and (t_4t_3). This is not true because t_3 must fire before t_4 according to the timing constraints of t_3 and t_4. As will be pointed out later, there is another issue related to the reachability graph if we merely use the absolute firing domain. This is why we use relative firing domains and absolute firing domains to determine the schedulability of individual transitions and the time span of transition sequences, respectively.

Similarly, $\delta_1 = (t_1t_2t_4t_3t_5)$ is not schedulable in Fig. 4a because t_4 cannot fire before t_3 according to $M_2 = \{p_3, p_4\}$, $D_2 = \{t_3(1,3), t_4(4,5)\}$), and

$$REFT_2(t_4) = 4 > MIN\{RLFT_2(t_3), RLFT_2(t_4)\} = 3.$$

Let us consider $\delta_2 = (t_1t_6t_7t_8)$. t_1 is schedulable and its firing results in $M_1 = \{p_2\}$ and $D_1 = \{t_2(1,4), t_6(5,7)\}$. Obviously, t_2 is schedulable and t_6 is not schedulable. Thus, δ_2 is not schedulable either. Note that $L(M_0, M_n) = \{\delta, \delta_1, \delta_2\}$, i.e., δ, δ_1, and δ_2 are exactly the three possible firing sequences reaching $M_n = \{p_7\}$. However, M_n is reachable only by means of δ. In Fig. 4b, the time Petri net has the same underlying Petri net as in Fig. 4a, but the static intervals of t_4 and t_6 are replaced with (2,4) and (3,7), respectively. In this case, δ, δ_1, and δ_2 are all schedulable. Their spans are (4,18), (4,17), and (8,24), respectively. The span of (t_3t_4) in δ is (2,4), whereas the span of (t_4t_3) in δ_1 is (2,3). It should also be noticed that the dynamic interval of t_6 in δ_2 is (3,4) and, therefore, the absolute interval of firing t_6 is (3,9).

4 Compositional Analysis of Schedulability

In this section, we describe how to conduct a schedulability analysis by decomposing a firing sequence in UN into a number of subsequences. Because of decomposition and composition, the analysis result of some sequence can be reused for checking other sequences. Specifically, the analysis of those sequences containing duplicated subsequences can be simplified. This not only reduces the complexity but also helps address the reachability issue.

As mentioned above, any marking in a schedule is stamped with an absolute time interval, relative to the initial marking. It is easy to get the time span between any two markings or transitions from a given schedule. To facilitate decomposition and composition, here, we use firing sequences instead of occurrence sequences and extend the schedulability analysis in the last section for more general cases. A sequence δ is allowed to start from any marking reachable from M_0 in UN, rather than M_0 itself. In other words, δ is allowed to be a part of a firing sequence. The algorithm of checking schedulability of δ is then denoted as a mapping $\Psi : S^T \rightarrow Q^* \times Q^*$, where S^T is the set of all firing (sub)sequences in UN. If δ is nonschedulable, then $\Psi(\delta)$ = (0,0); otherwise $\Psi(\delta) = TS_n$, which is the time span relative to the start time.

Definition 1. *Let* $\delta_1 = (M_{10}t_{11}M_{11} \ldots t_{1i}M_{1i} \ldots t_{1m}M_{1m})$ $(m \geq 1)$ *and* $\delta_2 = (M_{20}t_{21}M_{21} \ldots t_{2j}M_{2j} \ldots t_{2n}M_{2n})$ $(n \geq 1)$ *be two sequences in UN, where* M_{10} *and* M_{20} *are reachable from* M_0. δ_2 *is* composable *with* δ_1 *if and only if* $M_{1m} = M_{20}$ *and*

$EN(M_{1m}) \cap EN(M_{1m-1}) - \{t_{1m}\} = \emptyset$. *The composition of* δ_2 *with* δ_1*, denoted as* $\delta_1 + \delta_2$*, is*

$$(M_{10}t_{11}M_{11} \ldots t_{1i}M_{1i} \ldots t_{1m}M_{1m}t_{21}M_{21} \ldots t_{2j}M_{2j} \ldots t_{2n}M_{2n}).$$

$EN(M_{1m}) \cap EN(M_{1m-1}) - \{t_{1m}\} = \emptyset$ means that M_{1m} and M_{1m-1} do not share any other enabled transitions except t_{1m}, i.e., all transitions enabled by M_{1m} are newly enabled after firing t_{1m}. t_{1m}, if still enabled under M_{1m}, is considered as a new one. For example, in Fig. 4a, $(M_2t_3M_3t_4M_4t_5M_5)$ is composable with $(M_0t_1M_1t_2M_2)$ because $EN(M_2) = \{t_3, t_4\}$, $EN(M_1) = \{t_2, t_6\}$, and $EN(M_2) \cap EN(M_1) = \emptyset$; $(M_3t_4M_4t_5M_5)$ is not composable with $(M_0t_1M_1t_2M_2t_3M_3)$ because $EN(M_3) = \{t_4\}, EN(M_2) = \{t_3, t_4\}$ and $EN(M_3) \cap EN(M_2) \neq \emptyset$, i.e., $(t_1t_2t_3t_4t_5)$ cannot be decomposed into $(t_1t_2t_3)$ and (t_4t_5). Generally, it is incorrect to separate concurrent transitions while decomposing a sequence.

Obviously, sequence composition is associative, that is, $\delta_1 + \delta_2 + \delta_3 = (\delta_1 + \delta_2) + \delta_3 = \delta_1 + (\delta_2 + \delta_3)$, where δ_1, δ_2, and δ_3 are sequences. In the following, δ and δ_i are sequences, and $\delta_1 + \delta_2 + \ldots + \delta_k$ is simply denoted as $\delta_1\delta_2 \ldots \delta_k$.

Theorem 1. *Let* δ_2 *be composable with* δ_1. $\delta_1\delta_2$ *is schedulable if and only if both* δ_1 *and* δ_2 *are schedulable, and* $\Psi(\delta_1\delta_2) = \Psi(\delta_1) + \Psi(\delta_2)$ *if* $\delta_1\delta_2$ *is schedulable.*

Proof. Let

$$\delta_1 = (M_{10}t_{11}M_{11} \ldots t_{1i}M_{1i} \ldots t_{1m}M_{1m}),$$
$$\delta_2 = (M_{20}t_{21}M_{21} \ldots t_{2j}M_{2j} \ldots t_{2n}M_{2n}), \text{ and}$$
$$\delta_1\delta_2 = (M_{10}t_{11}M_{11} \ldots t_{1i}M_{1i} \ldots t_{1m}M_{1m}t_{21}M_{21} \ldots t_{2j}M_{2j} \ldots t_{2n}M_{2n}).$$

1. Suppose both δ_1 and δ_2 are schedulable. There exist two sequences of relative firing domains for checking the schedulability of δ_1 and δ_2, say, $(D_{10}D_{11} \ldots D_{1m})$, and $(D_{20}D_{21} \ldots D_{2n})$. Since δ_2 is composable with δ_1, $M_{1m} = M_{20}$, and

$$EN(M_{1m}) \cup EN(M_{1m-1}) - \{t_{1m}\} = \emptyset.$$

So, $D_{1m} = \{C(t) : t \in EN(M_{1m})\}$, i.e., for any transition enabled by M_{1m}, the dynamic interval is exactly the static interval. On the other hand, $D_{20} = \{C(t) : t \in EN(M_{20})\}$. Therefore, $D_{1m} = D_{20}$ and $(D_{10}D_{11} \ldots D_{1m}D_{21} \ldots D_{2n})$ is exactly the sequence of relative firing domains for checking the schedulability of $\delta_1\delta_2$. So, $\delta_1\delta_2$ is schedulable.

Similarly, there exist two sequences of time stamps for checking the schedulability of δ_1 and δ_2, say, $TS_{10}TS_{11} \ldots TS_{1m}$ and $TS_{20}TS_{21} \ldots TS_{2n}$, where $TS_{10} = TS_{20} = (0, 0)$. $\Psi(\delta_1) = TS_{1m}$ and $\Psi(\delta_2) = TS_{2n}$. There also exists two sequences of absolute firing domains for checking the schedulability of δ_1 and δ_2, say, $(AD_{10}AD_{11} \ldots AD_{1m})$ and $(AD_{20}AD_{21} \ldots AD_{2n})$. $AD_{1m} = \{C(t) + TS_{1m} : t \in EN(M_{20})\}$ and

$$AD_{20} = \{C(t) : t \in EN(M_{20})\}.$$

Let $AD_{2j}' = \{I + TS_{1m} : I \in AD_{2j}\}$ and $TS_{2j}' = TS_{2j} + TS_{1m}$ $(0 < j < n + 1)$. Then, $(AD_{10}AD_{11} \ldots AD_{1m}AD'_{21} \ldots AD'_{2n})$ is exactly the sequence of absolute firing domains and $(TS_{10}TS_{11} \ldots TS_{1m}TS'_{21} \ldots TS'_{2m})$ is exactly the sequence of time stamps for checking the schedulability of $\delta_1\delta_2$. So,

$$\Psi(\delta_1\delta_2) = TS'_{2n} = TS_{2n} + TS_{1m} = \Psi(\delta_2) + \Psi(\delta_1).$$

2. Suppose

$$\delta_1\delta_2 = (M_{10}t_{11}M_{11} \ldots t_{1i}M_{1i} \ldots t_{1m}M_{1m} t_{21}M_{21} \ldots t_{2j}M_{2j} \ldots t_{2n}M_{2n})$$

is schedulable. There exists a sequence of relative firing domains, say, $(D_{10}D_{11} \ldots D_{1m}D_{21} \ldots D_{2n})$. Obviously, $(D_{10}D_{11} \ldots D_{1m})$ and $(D_{1m}D_{21} \ldots D_{2n})$ are the sequences of relative firing domains for checking δ_1 and δ_2, respectively. So, both δ_1 and δ_2 are schedulable. Similarly, there exist a sequence of time stamps, say,

$$(TS_{10}TS_{11} \ldots TS_{1m}TS_{21} \ldots TS_{2n}).$$

$\Psi(\delta_1\delta_2) = TS_{2n}$. Obviously, $(TS_{10}TS_{11} \ldots TS_{1m})$ is the sequences of time stamps for checking δ_1. $\Psi(\delta_1) = TS_{1m}$. Suppose TS_{1m} doesn't contain infinite time. Let $TS_{20}' = (0, 0)$ and $TS_{2j}' = TS_{2j} - TS_{1m}(0 < j < n + 1)$. $TS_{20}'TS_{21}' \ldots TS_{2n}'$ is the sequence of time stamps for checking δ_2. $\Psi(\delta_2) = TS_{2n}' = TS_{2n} - TS_{1m}$. So,

$$\Psi(\delta_1) + \Psi(\delta_2) = TS_{1m} + TS_{2n} - TS_{1m} = TS_{2n} = \Psi(\delta_1\delta_2).$$

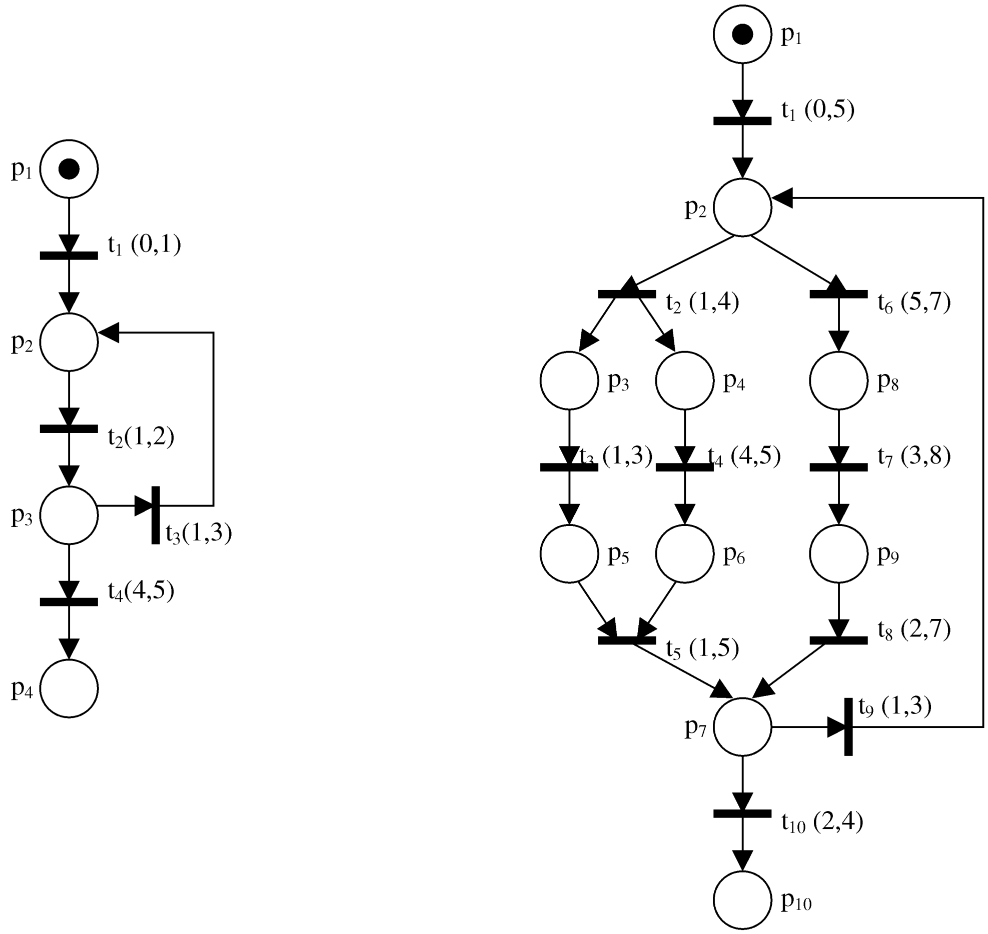
If TS_{1m} contains infinite time, $TS_{2j}(0 < j < n + 1)$ all contain infinite time. Similarly, we can prove $\Psi(\delta_1\delta_2) = \Psi(\delta_1) + \Psi(\delta_2)$.

According to 1 and 2, the theorem holds. □

If δ_2 is not composable with δ_1, checking δ_1 and δ_2 individually does not provide any useful information for analyzing the composition of δ_1 and δ_2. The reason is that the dynamic intervals in the initial relative and absolute firing domains are always equal to the static intervals. For example, let $\delta_1 = (M_0t_1M_1t_2M_2t_3M_3)$, $\delta_2 = (M_3t_4M_4t_5M_5)$, and $\delta = (M_0t_1M_1t_2M_2t_3M_3t_4M_4t_5M_5)$. In Fig. 4a, δ, δ_1, and δ_2 are all schedulable according to the extended algorithm, and $\Psi(\delta) = (6, 19)$, $\Psi(\delta_1) = (2, 12)$, $\Psi(\delta_2) = (5, 10)$. Obviously, $\Psi(\delta) \neq \Psi(\delta_1) + \Psi(\delta_2)$. As a result, we cannot analyze the schedulability of δ by means of δ_1 and δ_2.

Theorem 2. *Let* $\delta_i(1 \leq i \leq k)$ *be sequences, and* $\delta_i(2 \leq i \leq k)$ *be composable with* δ_{i-1}. $\delta_1 \ldots \delta_k$ *is schedulable if and only if* $\delta_i(1 \leq i \leq k)$ *are all schedulable.* $\Psi(\delta_1 \ldots \delta_k) = \sum_{i=1}^{k} \Psi(\delta_i)$ *if* $\delta_1 \ldots \delta_k$ *is schedulable.*

Proof. It is obvious if k = 2; if k = 3, then $\delta_1\delta_2\delta_3 = (\delta_1\delta_2)\delta_3$. Let $\delta = \delta_1\delta_2$. $\delta_1\delta_2\delta_3 = \delta\delta_3$. $\delta\delta_3$ is schedulable iff δ and δ_3 are schedulable iff δ_1, δ_2, and δ_3 are schedulable. Suppose $\delta = \delta_1 \ldots \delta_{k-1}$ is schedulable iff $\delta_i(1 \leq i \leq k - 1)$ are all schedulable. $\delta_1 \ldots \delta_k = \delta\delta_k$. $\delta_1 \ldots \delta_k$ is schedulable iff δ and δ_k are schedulable iff $\delta_i(1 \leq i \leq k)$ are schedulable. By induction, the theorem holds. □



(a) (b)

Fig. 5. Compositional analysis.

Theorem 2 shows the schedulability can be analyzed by decomposing a sequence into a number of subsequences, whenever possible and necessary.

Theorem 3. *Suppose sequence δ_2 is composable with itself (called self-composable) and with sequence δ_1 and sequence δ_3 is composable with δ_2. Let $\delta = (\delta_2)^k = \delta_2 \ldots \delta_2 \ldots \delta_2$, where the number of δ_2 is $k (k > 0)$. $\delta_1 \delta \delta_3$ is schedulable if and only if $\delta_1 \delta_2 \delta_3$ is schedulable. $\Psi(\delta_1 \delta \delta_3) = \Psi(\delta_1 \delta_2 \delta_3) + (k-1)^* \Psi(\delta_2)$ if $\delta_1 \delta \delta_3$ is schedulable.*

Proof. If $\delta_2 = (M_{20} t_{21} M_{21} \ldots t_{2i} M_{2i} \ldots t_{2m} M_{2m})$ is a self-composable sequence, then $M_{20} = M_{2m}$. This theorem directly follows from Theorem 2. □

The significance of Theorem 2 and Theorem 3 is that they not only simplify the schedulability analysis of those sequences containing loops, but also help analyze reachability. In Fig. 5a, there is an infinite number of firing (occurrence) sequences reaching $M_3 = \{p_4\}$ from $M_0 = \{p_1\}$ in the underlying net, but any firing sequence can be constructed by $(M_0 t_1 M_1 t_2 M_2)(\delta)^k (M_2 t_4 M_3)$, where $\delta = (M_2 t_3 M_1 t_2 M_2)$, $M_1 = \{p_2\}$, $M_2 = \{p_3\}$, and $k \in N$. According to Theorem 3, $(M_0 t_1 M_1 t_2 M_2)(\delta)^{100}(M_2 t_4 M_3)$ is schedulable if and only if $(M_0 t_1 M_1 t_2 M_2)\delta(M_2 t_4 M_3)$ is schedulable. Here, neither is schedulable because $(M_2 t_4 M_3)$ is nonschedulable. To determine the reachability of M_3 in TN, we only need to check the schedulability of basic sequences, i.e., $\delta_1 = (M_0 t_1 M_1 t_2 M_2 t_4 M_3)(k = 0)$ and $\delta_2 = (M_0 t_1 M_1 t_2 M_2 \delta M_2 t_4 M_3)(k = 1)$. With regard to timing constraints, M_3 is unreachable since neither δ_1 nor δ_2 is schedulable. In Fig. 5b, we can also identify some key transition sequences reaching $\{p_{10}\}$ in the underlying net, such as $(t_2 t_3 t_4 t_5)$, $(t_2 t_4 t_3 t_5)$, $(t_6 t_7 t_8)$, $(t_9 t_2 t_3 t_4 t_5)$, $(t_9 t_2 t_4 t_3 t_5)$, $(t_9 t_6 t_7 t_8)$, (t_1), and (t_{10}). Since $(t_2 t_4 t_3 t_5)$ and $(t_6 t_7 t_8)$ are nonschedulable from $M_1 = \{p_2\}$, any sequence containing $(t_2 t_4 t_3 t_5)$ or $(t_6 t_7 t_8)$ is nonschedulable. However, $(t_1)(t_2 t_3 t_4 t_5)(t_9 t_2 t_3 t_4 t_5)^k (t_{10})$ is schedulable, where $k \in N$. Thus, $\{p_{10}\}$ is reachable in the TN. Note that Theorems 1-3 are useful only if sequences can be decomposed, i.e., no concurrency is present at decomposition markings.

5 Schedulability and Reachability

In this section, we discuss the reachability problem of TPNs according to the reachability graph of PNs. Though we know M_n is reachable by finding a schedule that starts from M_0 to M_n, it is generally difficult to determine whether M_n is reachable or when the reachable marking M_n is reached because all possible firing sequences from M_0 to M_n in a UN (i.e., in $L(M_0, M_n)$) must be analyzed.

Figs. 4a and 4b have the same underlying Petri net. $L(M_0, M_n) = \{\delta = (t_1t_2t_3t_4t_5), \delta_1 = (t_1t_2t_4t_3t_5), \delta_2 = (t_1t_6t_7t_8)\}$, where $M_n = \{p_7\}$. In Fig. 4a, δ is schedulable, and $\Psi(\delta) = (6, 19)$, whereas δ_1 and δ_2 are nonschedulable. The earliest time and the latest time when M_n is reached are (6, 19). In Fig. 4b, all δ, δ_1, and δ_2 are schedulable. $\Psi(\delta) = (4, 18)$, $\Psi(\delta_1) = (4, 17)$, and $\Psi(\delta_2) = (8, 24)$. Therefore, the earliest time and the latest time when M_n is reached are (4, 24).

However, can we determine whether M_n is unreachable if none of δ, δ_1, or δ_2 are schedulable? It is known that the reachability and boundedness problems are decidable for PNs [9], but undecidable for TPNs [1]. Even if M_n is reachable in a UN, it may not be reachable in a TN. This does not mean that we cannot establish some relationship between the reachability of a TN and the reachability of its UN. In fact, we have:

Theorem 4. *M_n is unreachable in a TN if marking M_n is unreachable in its UN.*

Proof. We need to show that, if M_n is reachable in a TN, then M_n is reachable in its UN. If M_n is reachable in a TN, there exists a firing schedule, say

$$< M_0, D_0 > \xrightarrow{t_1(\theta_1)} < M_1, D_1 >_{\dots} \xrightarrow{t_i(\theta_i)}$$
$$< M_i, D_i > \dots \xrightarrow{t_n(\theta_n)} < M_n, D_n > .$$

t_1 is firable and schedulable by $< M_0, D_0 >$ in the TN, so t_1 in the UN is firable by M_0 and the firing of t_1 in the UN reaches M_1 exactly. Therefore, $(M_0t_1M_1)$ is a firing sequence in the UN. By induction,

$$(M_0t_1M_1 \dots t_iM_i \dots t_nM_n)$$

is a firing sequence in the UN, that is, M_n is reachable in the UN. □

Theorem 5. *It is decidable whether M_n in a TN is reachable or not if $L(M_0, M_n)$ is a finite set.*

Proof. Since $L(M_0, M_n)$ is finite, there is a finite number of firing sequences that reach M_n in the UN. For any firing sequence δ in $L(M_0, M_n)$, we can check whether δ is schedulable or not in the TN. If there exists at least one schedulable firing sequence, then M_n is schedulable in the TN, otherwise M_n is nonschedulable in the TN according to Theorem 4. □

If M_n is schedulable in a TN, the earliest time when M_n is reached is the minimum of the times of schedulable firing sequences reaching M_n, i.e., $MIN\{AE(\delta) : \delta \in L(M_0, M_n) \wedge \delta$ is schedulable$\}$ and the latest time is $MAX\{AL(\delta) : \delta \in L(M_0, M_n) \wedge \delta$ is schedulable$\}$, where $(AE(\delta), AL(\delta)) = \Psi(\delta)$. Theorem 5 is less practical because a finite $L(M_0, M_n)$ means there is no loop from M_0 to M_n in the net. Now, we extend it to infinite $L(M_0, M_n)$ where sequences can be composed.

Definition 2. *A Petri net $UN = (P, T, B, F, M_0)$ is said to be well structured with respect to a reachable marking M_n if there exists a finite set S of composable sequences such that any firing sequence in $L(M_0, M_n)$ can be composed from S.*

For example,

$$S = \{(M_0t_1M_1t_2M_2), (M_2t_4M_3), (M_2t_3M_1t_2M_2)\}$$

for the underlying net in Fig. 5a and $M_n = M_3 = \{p_4\}$ and

$$\begin{aligned} S = \{&(M_0t_1M_1), (M_1t_2t_3t_4t_5M_5), (M_1t_2t_4t_3t_5M_5), \\ &(M_1t_6t_7t_8M_5), (M_5t_9t_2t_3t_4t_5M_5), (M_5t_9t_2t_4t_3t_5M_5), \\ &(M_5t_9t_6t_7t_8M_5), (M_5t_{10}M_n)\} \end{aligned}$$

for the underlying net in Fig. 5b and $M_n = \{p_{10}\}$. Thus, the nets in Fig. 5 are well structured with respect to M_n.

In practice, whether a UN net is well structured with respect to M_n can be determined in terms of the reachability tree of the UN. The UN is a well-structured if 1) $BL(M_0, M_n)$ is finite, i.e., there is a finite number of basic firing sequences (without duplicate subsequences). 2) For any basic loop $(M_jt_kM_k \dots M_{l-1}t_lM_i \dots .t_jM_j)$ (without duplicate subsequences) in path $(M_0t_1M_1 \dots t_iM_i \dots t_jM_j \dots t_nM_n)$, $(M_jt_kM_k)$ is composable with $(t_iM_i \dots .t_jM_j)$ and $(M_it_{i+1}M_{i+1})$ is composable with $(t_kM_k \dots t_lM_i)$. That is,

$$UN(M_j) \cap UN(M_{j-1}) - \{t_j\} = \emptyset \text{ and}$$
$$UN(M_{l-1}) \cap UN(M_i) - \{t_l\} = \emptyset.$$

Such basic loops are called *composable loops*. In addition, a UN net is not well structured if concurrent branches have internal loops (see the end of the next section).

For the underlying net in Fig. 5a and $M_n = M_3 = \{p_4\}$, $BL(M_0, M_n) = \{(M_0t_1M_1t_2M_2t_4M_3)\}$ and loop $(M_2t_3M_1t_2M_2)$ $(i = 1, j = 2, t_l = t_k = t_3, \text{ and } M_{l-1} = M_j)$ is a composable loop because $UN(M_j) \cap UN(M_{j-1}) = UN(M_{l-1}) \cap UN(M_i) = UN(M_2) \cap UN(M_1) = \{t_3, t_4\} \cap \{t_2\} = \emptyset$. Similarly, for the underlying net in Fig. 5b and $M_n = \{p_{10}\}$,

$$BL(M_0, M_n) = \{(M_0t_1t_2t_3t_4t_5t_{10}), (t_1t_2t_4t_3t_5t_{10}), (t_1t_6t_7t_8t_{10})\},$$

and basic loops $(t_9t_2t_3t_4t_5), (t_9t_2t_4t_3t_5), (t_9t_6t_7t_8)$ are all composable.

Theorem 6. *The reachability of M_n in a TN is decidable if the UN is well structured with respect to M_n.*

Proof. Suppose the UN is well structured with respect to M_n. There exists a finite set of composable sequences S such that any firing sequence in $L(M_0, M_n)$ can be composed from S. In terms of S, we can build a set of basic firing sequences reach M_n from M_0, say, $BL(M_0, M_n) \subseteq L(M_0, M_n)$, such that, for any $\delta \in BL(M_0, M_n)$, δ cannot be decomposed into $\delta_1..\delta_i\delta_i \dots \delta_k$, where $\delta_i(i = 1 \dots k) \in S$. That is, δ does not contain duplicate composable sequences. Since S is finite, $BL(M_0, M_n)$ is finite too. According to Theorem 2 and Theorem 3, if none of firing sequences in $BL(M_0, M_n)$ is schedulable, then none of firing sequences in $L(M_0, M_n)$ is schedulable, i.e., M_n is unreachable in the TN. If there is a schedulable firing sequence in $BL(M_0, M_n)$, M_n is reachable in the TN. In this case, the earliest time when M_n is reached is $MIN\{AE(\delta) : \delta \in BL(M_0, M_n) \wedge \delta$ is schedulable$\}$. Thus, whether M_n is reachable is decidable. □

According to Theorem 6, the reachability of $M_n = \{p_{10}\}$ in Fig. 5b is decidable no matter what timing constraints are imposed on the well structured underlying net. Thus,

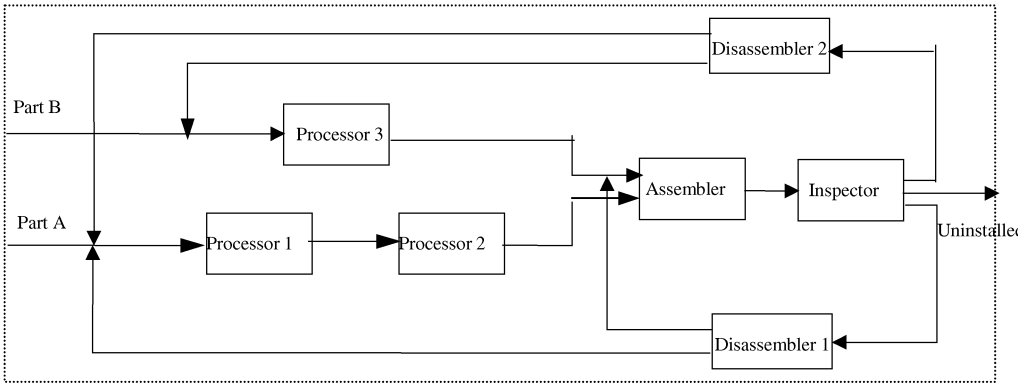


Fig. 6. The architecture of the assembly system.

Theorem 6 is more general than Theorem 5 because the underlying net is well structured if $L(M_0, M_n)$ is a finite set. It should be noticed that the schedulability analysis in this paper is not intended to address the general reachability problem though they are closely related. As a matter of fact, $< M_i, D_i >$ is a state class in the enumerative method [1] and schedule

$$< M_0, D_0 > \xrightarrow{t_1} < M_1, D_1 > \ldots$$
$$\xrightarrow{t_i} < M_i, D_i > \ldots \xrightarrow{t_n} < M_n, D_n >$$

is a path in the reachability graph. The time span of executing such a schedule can be evaluated according to the absolute firing domains. Nevertheless, the values in a sequence of absolute firing domains and in a sequence of time stamps are monotonically increased. They cannot be used as a part of state classes to generate the reachability graph that contains loops, otherwise it makes no sense to compare two state classes. For example, two state classes with the same marking and same relative firing domain reached by different schedules are usually unequal since they have different absolute firing domains and time stamps. This reflects a reason why absolute time mode is used to conduct the schedulability analysis, rather than to address the general reachability issue, for TPNs.

6 Schedulability Analysis of an Assembly System

We have applied SAM to model various time-dependent software system architectures, such as flexible manufacturing systems (FMS) [12] and command and control systems (C2) [13], and used the method described in this paper to conduct schedulability analysis of these system architectures. This section describes how to analyze the schedulability of an assembly subsystem in an FMS, separating timing properties from functional properties.

FMS systems provide a means to achieve better quality, lower cost, and smaller lead-time in manufacturing. An FMS is a real-time system composed of a number of computer-controlled tools and automated material handling, assembly, and storage systems that operate as an integrated system under the control of host computers. The growing demand for higher performance and flexibility in these systems and the interlocking factors of concurrency, deadline-driven activities, and real-time decision-making pose a significant challenge to FMS design, especially in terms of control and scheduling. For a complex FMS, it is necessary to experiment with different alternatives of control and scheduling policies against the same hardware configuration. It is therefore highly desirable to be able to "plug-in" the specifications of various control modules to an FMS model without having to make major changes or reconstruct the entire system model each time.

Let us consider the assembly subsystem in an FMS. As shown in Fig. 6, the assembly system is composed of three processors, one inspector, one assembler, and two disassemblers. The system receives two types of parts (A and B) as inputs and, after processing the input parts, one A-part and one B-part are assembled into a final product. The assembly procedure is described as follows: raw parts arrive in pairs, A-parts are processed by processors 1 and 2 in series, while B-parts are processed by processor 3. Processed A-parts and B-parts are finally assembled by an assembler. The inspector is responsible for quality control of the assembled products. If an assembled product satisfies the quality requirements, it is unloaded from the system as a final product; otherwise, it is disassembled either by disassembler 1 or 2 depending upon their status. Disassembler 1 generates A-parts to be sent back to processors 1 and 2 and B-parts to assembler, respectively. Disassembler 2 generates A-parts to be sent back to processors 1 and 2 and B-parts to processor 3. For all processors, inspectors, assemblers, and disassemblers, there are certain timing constraints imposed on them (refer to Table 1). Considering timing constraints, we need to check a number of assembling schedules. Whether an A-part and a B-part are assembled in a given time period mostly depends on the analysis of following cases:

1. Neither A-part nor B-part has any quality problem.
2. There is no problem with B-part, but A-part cannot pass the quality examination for m times ($m > 0$).
3. A-part and B-part cannot pass the quality examination for m and n times respectively ($m,n > 0$).

The TPN model of the assembly system is shown in Fig. 7. The places and transitions are described in Table 1. Suppose the assembly system receives an A-part and a B-part at sometime, that is, $M_0 = \{p_{i1}, p_{i2}\}$. Without

TABLE 1
Legend for Fig. 5

Place	Description	
p_{i1},p_{i2}	Input from other subsystems providing A-parts and B-parts	
p_o	Output to another subsystem storing assembled products	
p_1	A-part	
p_2	A-part after the processing of processor 1	
p_3	A-part for assembly after the processing of processor 2	
p_4	B-part	
p_5	B-part for assembly	
p_6	Assembled product	
p_7	Examining result of inspector	
Transition	**Description**	**Interval**
t_1	Receive A-part and B-part	(0,1)
t_2	Processor 1 works on an A-part	(1,3)
t_3	Processor 2 works on an A-part	(1,2)
t_4	Processor 3 works on a B-part	(2,4)
t_5	Assembler works	(1,2)
t_6	Inspector examines assembled product	(0,1)
t_7	Final product is unloaded	(0,2)
t_8	Disassembler 1 works	(0,1)
t_9	Disassembler 2 works	(0,1)

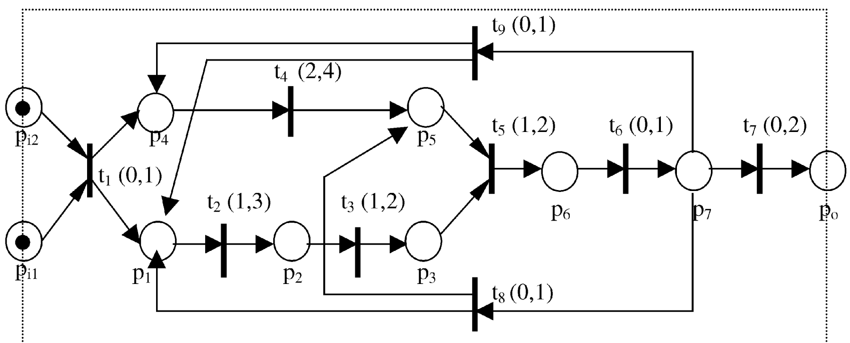


Fig. 7. TPN model of the assembly system.

consideration of timing constraints, the functional requirements of above cases are easily analyzed. For example,

$$\delta_1 = (t_1t_2t_3t_4t_5t_6t_7),$$
$$\delta_2 = (t_1t_2t_3t_4t_5t_6t_8t_2t_3t_5t_6t_7), \text{ and}$$
$$\delta_3 = (t_1t_2t_3t_4t_5t_6t_9t_2t_3t_4t_5t_6t_7)$$

are three basic occurrence sequences in the underlying net that model the functional behaviors of correspondent assembly processes.

According to the algorithm in Section 2, the schedulability of δ_1 is analyzed as follows:

1. $TS_0 = (0,0)$; $//M_0 = \{p_{i1}, p_{i2}\}$;
   $D_0 = \{t_1(0,1)\}$; $AD_0 = \{t_1(0,1)\}$.
2. t_1 is schedulable during (0,1); $//M_1 = \{p_1, p_4\}$;
   $TS_1 = (0,1)$; $D_1 = \{t_2(1,3), t_4(2,4)\}$;
   $AD_1 = \{t_2(1,4), t_4(2,5)\}$.
3. t_2 is schedulable during (1,3); $//M_2 = \{p_2, p_4\}$;
   $TS_2 = (1,4)$; $D_2 = \{t_3(1,2), t_4(0,3)\}$;
   $AD_2 = \{t_3(2,6), t_4(2,5)\}$.
4. t_3 is schedulable during (1,2); $//M_3 = \{p_3, p_4\}$;
   $TS_3 = (2,5)$; $D_3 = \{t_4(0,2)\}$; $AD_3 = \{t_4(2,5)\}$.
5. t_4 is schedulable during (0,2); $//M_4 = \{p_3, p_5\}$;
   $TS_4 = (2,5)$; $D_4 = \{t_5(1,3)\}$; $AD_4 = \{t_5(3,8)\}$.
6. t_5 is schedulable during (1,5); $//M_5 = \{p_6\}$;
   $TS_5 = (3,8)$; $D_5 = \{t_6(0,1)\}$; $AD_5 = \{t_6(3,9)\}$.
7. t_6 is schedulable during (0,1); $//M_6 = \{p_7\}$;
   $TS_6 = (3,9)$; $D_6 = \{t_7(0,2), t_8(0,1), t_7(0,1)\}$;
   $AD_6 = \{t_7(3,11), t_8(3,10), t_7(3,10)\}$.
8. t_7 is schedulable during (0,1); $//M_7 = \{p_0\}$;
   $TS_7 = (3,10)$; $D_7 = \emptyset$; $AD_7 = \emptyset$.

Thus, δ_1 is schedulable, and $\Psi(\delta_1) = (3,10)$. Similarly, δ_2 and δ_3 are also schedulable, and $\Psi(\delta_2) = (6,19)$ and $\Psi(\delta_3) = (6,18)$.

Besides the analysis of basic sequences, we also need to deal with concurrence and loop. Here, t_4 and t_2t_3 are concurrent. $(t_2t_3t_4)$, $(t_2t_4t_3)$, and $(t_4t_2t_3)$ are all schedulable from marking $M_1 = \{p_1, p_4\}$. The sequences obtained from δ_1, δ_2, and δ_3 by replacing $(t_2t_3t_4)$ with $(t_2t_4t_3)$ or $(t_4t_2t_3)$ are all schedulable. For example, if $(t_2t_3t_4)$ in δ_1 is replaced with $(t_4t_2t_3)$, we can change steps 3, 4, and 5 as follows:

3. t_4 is schedulable during (2,3); $//M_2 = \{p_1, p_5\}$;
   $TS_2 = (2,4)$; $D_2 = \{t_2(0,1)\}$; $AD_2 = \{t_2(2,4)\}$.
4. t_2 is schedulable during (0,1); $//M_3 = \{p_2, p_5\}$;
   $TS_3 = (2,4)$; $D_3 = \{t_3(1,2)\}$; $AD_3 = \{t_3(3,6)\}$.

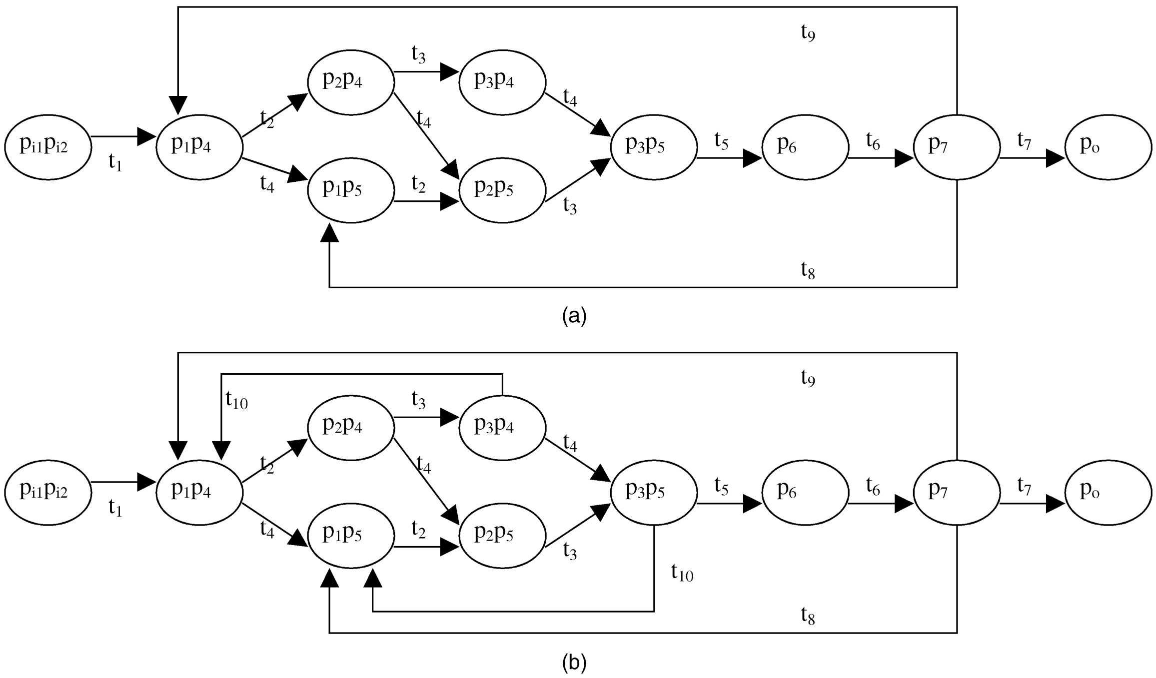


Fig. 8. Reachability trees of underlying nets.

5. t_3 is schedulable during (1,2); $//M_4 = \{p_3, p_5\}$;
$TS_4 = (3,6)$; $D_4 = \{t_5(1,3)\}$; $AD_4 = \{t_5(4,9)\}$.

If $(t_2t_3t_4)$ in δ_1 is replaced with $(t_2t_4t_3)$, we can change steps (3), (4), and (5) as follows:

3. t_2 is schedulable during (1,2); $//M_2 = \{p_2, p_4\}$;
$TS_2 = (1,4)$; $D_2 = \{t_3(1,2), t_4(0,3)\}$;
$AD_2 = \{t_3(2,6), t_4(2,5)\}$.
4. t_4 is schedulable during (0,2); $//M_3 = \{p_2, p_5\}$;
$TS_3 = (2,5)$; $D_3 = \{t_3(0,2)\}$; $AD_3 = \{t_3(2,6)\}$.
5. t_3 is schedulable during (0,2); $//M_4 = \{p_3, p_5\}$;
$TS_4 = (2,6)$; $D_4 = \{t_5(1,3)\}$; $AD_4 = \{t_5(3,9)\}$.

$\Psi(t_2t_3t_4)$, $\Psi(t_2t_4t_3)$, and $\Psi(t_4t_2t_3)$ relative to the time at M_1 are (2,4), (2,5), and (3,5), respectively. The loops that should be taken into account are $\sigma_1 = (t_8t_2t_3t_5t_6)$ and $\sigma_2 = (t_9\sigma t_5t_6)$, where $\sigma = (t_2t_3t_4)$ or $(t_2t_4t_3)$, or $(t_4t_2t_3)$. It is easy to show that $\Psi(\sigma_1) = (3,9)$ and $\Psi(\sigma_2) = \Psi(\sigma) + (1,4)$, relative to the time at $M_6 = \{p_7\}$.

Recall the assembling schedules. They are actually correspondent to the following task sequences:

1. $(t_1\sigma t_5t_6t_7)$,
2. $(t_1\sigma t_5t_6\sigma_1^m t_7)$, and
3. $(t_1\sigma t_5t_6\sigma_1^m\sigma_2^n t_7)$,

where (m > 0 and n > 0). These sequences are obviously occurrence sequences in the underlying net and their timing properties are easily calculated according to above discussion. Note that σ_1 and σ_2 are self-composable sequences, and σ_2 (σ_1) is composable with σ_1 (σ_2). Both the starting marking and the ending marking are $\{p_7\}$. $\sigma_1\sigma_2 = \sigma_2\sigma_1$ (i.e., the ordering of σ_1 and σ_2 is not important). So, A-part fails m times and B-part fails n times at any sequences can be represented by $\sigma_1^m\sigma_2^n$. Moreover, the underlying net is well structured with respect to $M_n = \{p_o\}$ because any firing sequences can be composed from $S = \{t_1\sigma t_5t_6, \sigma_1, \sigma_2, t_7\}$ according to the reachability tree in Fig. 8a. The set of basic sequences $BL(M_0, M_n) = \{t_1\sigma t_5t_6t_7\}$ and both basic loops σ_2 and σ_1 are composable. So, we can determine whether M_n is reachable and whether a given task execution is schedulable no matter what the timing constraints are. For the constraints given in Fig. 7, there is no nonschedulable transition and M_n is reachable. However, if we add a new transition t_{10} with interval (0,2), input place p_3, and output place p_2 to Fig. 7, then the new underlying net is not well structured (branch t_2t_3 with a loop $t_{10}t_2t_3$ is concurrent with branch t_4). The reachability tree of the new underlying net is shown in Fig. 8b. Basic loop $t_{10}t_2t_3$ is not composable because

$$\begin{aligned} & UN(\{p_3p_4\}) \cap UN(\{p_2p_4\}) - \{t_3\} \\ & = \{t_{10}, t_4\} \cap \{t_3, t_4\} - \{t_3\} = \{t_4\} \neq \emptyset, \end{aligned}$$

where $\{p_3p_4\}$ and $\{p_2p_4\}$ are the markings after/before the firing of t_3 in the loop. To determine the schedulability of a sequence with such loops, we cannot use the compositional technique described in this paper.

7 Conclusions

We have presented an approach to the schedulability analysis of real-time systems modeled by time Petri nets. The contribution of this paper includes:

1. An approach for schedulability analysis by separating timing properties from other behavioral properties and by using relative/absolute time modes to determine the schedulability of individual transitions and to evaluate the time span of task execution. This provides an incremental verification technique from Petri nets to time Petri nets in the software architecture methodology SAM.

2. A compositional technique to reduce the complexity of schedulability analysis by decomposing a complicated task execution into a number of subsequences.
3. A relationship between some reachability and timing issues of time Petri nets and the reachability of underlying Petri nets and the compositional analysis.
4. Identification of a class of time Petri nets with well-structured underlying Petri nets so that the reachability of these nets can be easily analyzed.

Our compositional schedulability analysis is applicable to TPNs that model behaviors and timing constraints of individual system components (subsystems) and connections (communication and interaction among components) of real time systems. An interesting research problem is how the approach can be extended to analyze distributed real time systems. In particular, how to compose system components modeled by TPNs at a higher abstraction level is our on-going research problem.

ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation under grants HDR-9707076 and CCR-0098120, and by NASA under grant NAG 2-1440. The authors are grateful to Dr. Jiacun Wang for his earlier discussion on this work. The authors also thank the anonymous reviewers for their comments, which have helped us to improve this manuscript.

REFERENCES

[1] B. Berthomieu and M. Diaz, "Modeling and Verification of Time Dependent Systems Using Time Petri Nets," *IEEE Trans. Software Eng.*, vol. 17, no. 3, pp. 259-273, Mar. 1991.
[2] G. Bucci and E. Vicario, "Compositional Validation of Time-Critical Systems Using Communicating Time Petri Nets," *IEEE Trans. Software Eng.*, vol. 21, no. 12, pp. 969-992, Dec. 1995.
[3] M. Felder, D. Mandrioli, and A. Morzenti, "Proving Properties of Real Time Systems through Logical Specifications and Petri Net Models," *IEEE Trans. Software Eng.*, vol. 20, no. 2, pp. 127-141, Feb. 1994.
[4] C. Ghezzi, D. Mandrioli, S. Morasca, and M. Pezze, "A Unified High-Level Petri Net Formalism for Time Critical Systems," *IEEE Trans. Software Eng.*, vol. 17, no. 2, pp. 160-171, Feb. 1991.
[5] D. Haban and K.G. Shin, "Application of Real-Time Monitoring to Scheduling Tasks with Random Execution Times," *IEEE Trans. Software Eng.*, vol. 16, no. 12, pp. 1374-1389, Dec. 1990.
[6] F. Jahanian and A.K.-L. Mok, "Safety Analysis of Timing Properties in Real-Time Systems," *IEEE Trans. Software Eng.*, vol. 12, pp. 890-904, 1986.
[7] E.Y.T. Juan, J.J.P. Tsai, and T. Murata, "Compositional Verification of Concurrent Systems Using Petri-Net-Based Condensation Rules," *ACM Trans. Programming Languages and Systems*, vol. 21, no. 5, pp. 917-979, 1998.
[8] P.M. Merlin and D.J. Farber, "Recoverability of Communication Protocols," *IEEE Trans. Comm.*, vol. 24, no. 4, pp. 1036-1043, 1976.
[9] T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541-580, Apr. 1989.
[10] A.D. Stoyenko, C. Hamacher, and R.C. Holt, "Analyzing Hard-Real-Time Programs for Guaranteed Schedulability," *IEEE Trans. Software Eng.*, vol. 17, no. 8, pp. 737-750, Aug. 1991.
[11] J.J.P. Tsai, S.J. Yang, and Y.-H. Chang, "Timing Constraint Petri Nets and Their Application to Schedulability Analysis of Real-Time System Specifications," *IEEE Trans. Software Eng.*, vol. 21, no. 1, pp. 32-49, Jan. 1995.
[12] J. Wang and Y. Deng, "Incremental Modeling and Verification of Flexible Manufacturing Systems," *J. Intelligent Manufacturing*, no. 4, 1999.
[13] J. Wang, X. He, and Y. Deng, "Software Architecture Specification and Analysis in SAM: A Case Study," *Information and Software Technology*, vol. 41, no. 7, pp. 451-467, 1999.
[14] J. Wang, G. Xu, and Y. Deng, "Reachability Analysis of Real-Time Systems Using Time Petri Nets," *IEEE Trans. Systems, Man, and Cybernatics*, vol. 30, no. 5, Oct. 2000.

Dianxiang Xu received the BS, MS, and PhD degrees in computer science from Nanjing University, China in 1989, 1992, and 1995, respectively. He joined the department of computer science, Texas A&M University in August 2000, and is currently a research assistant professor. From May 1999 to August 2000, he was a research associate at the school of computer science, Florida International University. Prior to that, he was an associate professor and an associate head of the department of computer science and technology, Nanjing University. Dr. Xu has authored and coauthored more than 30 publications in international journals and conference proceedings. His research interests are in the areas of Petri nets, multiagent teamwork, mobile agents, distributed system engineering, and knowledge-based systems. He is a member of the IEEE Computer Society.

Xudong He received the BS and MS degrees in computer science from Nanjing University, China, in 1982 and 1984, respectively. He received the PhD degree in computer science from Virginia Polytechnic Institute and State University (Virginia Tech), Blacksburg, in 1989. He joined the faculty in the School of Computer Science at Florida International University (FIU) in 2000, and is an associate professor of the School of Computer Science and the director of the Center for Advanced Distributed System Engineering. Prior to joining FIU, he was an associate professor in the Department of Computer Science at North Dakota State University. His research interests include formal methods, especially Petri nets, and software testing techniques. He has published more than 50 papers in the above areas. He is a member of the Association for Computing Machinery and the IEEE Computer Society.

Yi Deng received the PhD degree in computer science from the University of Pittsburgh in 1992. He is the director of the School of Computer Science at the Florida International University (FIU)—The State University of Florida at Miami. From August 2000 to May 2002, he was with the University of Texas at Dallas (UT-Dallas) as the Managing Director of the Embedded Software Center and an associate professor of computer science in the School of Engineering and Computer Science. Prior to joining UT-Dallas, he was an associate professor of computer science at FIU and the founding Director of the Center for Advanced Distributed Systems Engineering (CADSE)—a university research center designated by the Florida Board of Regents. His research interests include component-based software engineering, software architecture, formal methods for complex systems, modeling and analysis of software security systems and middleware. He has published more than 60 papers in various journals and conferences. He has been the principal investigator (PI) or Co-PI of 14 research awards of over seven million dollars from the US National Science Foundation, US Air Force Office of Scientific Research (AFOSR), US Army Research Office (ARO), US Air Force Rome Laboratory, and the National Aeronautics and Space Administration (NASA), as well as from industry. He is an editor for the *International Journal of Software Engineering and Knowledge Engineering* and has been the program committee chair or member for several conferences. He is a member of the ACM, IEEE, and IEEE Computer Society.