Checking MSC Specifications for Timing Inconsistency

LI Xuandong, TAN Wenkai, and ZHENG Guoliang

State Key Laboratory for Novel Software Technology, Department of Computer Science and Technology
Nanjing University, Nanjing 210093, P.R. China
E-mail: lxd@nju.edu.cn
Received December 15, 1999; revised June 26, 2000.

Abstract Message sequence chart (MSC) is a graphical and textual language for the description and specification of the interactions between system components. MSC specifications allow convenient expression of multiple scenarios, and offer an intuitive and visual way of describing design requirements. Like any other aspect of the specification and design process, MSCs are amenable to errors, and their analysis is important. In this paper, the verification problem of MSC specification for timing inconsistency is studied, which means that no execution scenario described by an MSC specification is timing consistent. An algorithm is developed to check MSC specifications for timing inconsistency.

Keywords real-time system, message sequence chart, model checking, timing inconsistency

1 Introduction

Message sequence chart (MSC) is a graphical and textual language for the description and specification of the interactions between system components. The main area of application for MSCs is in the overview specification of the communication behavior of real-time systems, in particular telecommunication switching systems. Such specifications are scenario-based and focus on message exchanges among communicating entities in distributed software systems. They offer an intuitive and visual way of describing design requirements.

MSCs represent typical execution scenarios, providing examples of either normal or exceptional executions of the proposed system. The MSC standard defined by ITU-T in Recommendation Z.120[1] introduces two basic concepts: basic MSCs and high-level MSCs. A basic MSC describes exactly one scenario. For describing multiple scenarios, MSC specifications are introduced, which are a combination of a set of basic MSCs and a high-level MSC describing their compositions. For example, Fig.1 depicts a simple MSC specification.

Like any other aspect of the specification and design process, MSCs are amenable to errors, and their analysis is important. A comprehensive study of the model checking of MSCs for temporal requirements has been given in [2]. For MSCs with timing constraints used to describe real-time systems, the verification problems are more complicated. In [3], some algorithms for analyzing basic MSCs with interval delays are presented, and a corresponding tool is described. In [4, 5], timing analysis is extended to checking MSC specifications for timing consistency, which means that every execution scenario described by an MSC specification is timing consistent. In this paper, we consider the problem of checking MSC specifications for timing inconsistency, which means that no execution scenario described by an MSC specification is timing consistent. For example, for the MSC specification depicted in Fig.1, if there are timing constraints that the separation time between sending the message $m_1$ (denoted as $e_1$) and receiving the message $m_6$ (denoted as $e_2$) is not smaller than 25 time units, which is denoted as $e_{12} - e_1 \geq 25$, and that the separation time between sending the message $m_5$ (denoted as $e_5$) and receiving the message $m_4$ (denoted as $e_5$) is not smaller than 34 time units, which is denoted as $e_8 - e_5 \geq 34$, then the MSC specification is timing consistent. But if the

This work is supported by the National Natural Science Foundation of China (No. 60073031 and No. 69703009), the National ‘863’ High-Tech Programme of China (No. 863-306-ZT04-4) and Jiangsu Province Research Foundation.
timing constraints are changed into $25 \leq e_{12} - e_1 \leq 30$ and $e_8 - e_5 \geq 34$, then the MSC specification is timing inconsistent.

![MSC Diagram](image)

Fig. 1. A simple MSC specification.

In this paper, we solve the verification problem by developing an algorithm to check MSC specifications for timing inconsistency. The paper is organized as follows. In the next section, we introduce MSC specifications. Section 3 gives an algorithm to check MSC specifications for timing inconsistency. The last section is the conclusion of the paper.

## 2 MSC Specifications

### 2.1 Basic MSCs

The MSC standard defined by ITU-T in Recommendation Z.120\[1] introduces two basic concepts: basic MSCs (bMSCs) and high-level MSCs (hMSCs). A bMSC consists of a set of processes that run in parallel and exchange messages in a one-to-one, asynchronous fashion. An hMSC graphically combines references with bMSCs to describe parallel, sequential, iterative, and non-deterministic executions of the bMSCs.

The semantics of an MSC essentially consists of sequences (of traces) of messages that are sent and received among the concurrent processes in the MSC. The order of communication events (i.e., message sending or receiving) in a trace is deduced from the visual partial order determined by the flow of control within each process in the MSC along with a causal dependency between the events of sending and receiving a message\[6].

For facilitating the specifications of real-time systems, several mechanisms have been introduced to describe timing constraints in MSCs, which are timers\[1], interval delays\[2], and timing marks\[7]. In this paper, we consider MSCs with more general timing constraints. In an MSC, by events we mean message sending and message receiving. Each event is given a name which represents its occurrence time. So, timing constraints can be described by Boolean expressions on event names. Here we let any timing constraint be of the form

$$a \leq e_0(e_0 - e'_0) + e_1(e_1 - e'_1) + \cdots + e_n(e_n - e'_n) \leq b,$$

where $e_0, e'_0, e_1, e'_1, \ldots, e_n, e'_n$ are event names, $a, b$ and $e_0, e_1, \ldots, e_n$ are real numbers ($b$ may be $\infty$). For example, for the hMSC $M_3$ in Fig.1, if it is enforced by a timing constraint that the separation time between sending the message $m_3$ (denoted as $e_5$) and receiving the message $m_4$ (denoted as $e_8$) is not smaller than 34 time units, the timing constraint can be expressed by $e_8 - e_5 \geq 34$.

For analyzing MSCs, we formalize bMSCs as follows.

**Definition 1.** A bMSC is a tuple $M = (P,E,V,C)$ where
• \( P \) is a finite set of processes;
• \( E \) is a finite set of events corresponding to sending a message and receiving a message;
• \( \mathcal{V} \) is a finite set whose elements are of the form \((e,e')\), where \( e \) and \( e' \) are in \( E \) and \( e' \neq e \), which represents a visual order displayed in \( M \);
• \( C \) is a set of Boolean expressions, which represents the timing constraints enforced on \( M \).

For example, the bMSC \( M_2 \) in Fig.1 can be represented by the tuple \((P,E,\mathcal{V},C)\) where

\[
\begin{align*}
P &= \{P_1, P_2, P_3\}, \\
E &= \{e_5, e_6, e_7, e_8\}, \\
\mathcal{V} &= \{(e_5, e_6), (e_6, e_7), (e_7, e_8)\}, \\
C &= \{e_8 - e_5 \geq 34\}.
\end{align*}
\]

We use event sequences to represent the untimed behaviour of bMSCs. Any event sequence is of the form \( e_0 e_1 \ldots e_m \), which represents that \( e_{i+1} \) takes place after \( e_i \) for any \( i \) \( (0 \leq i \leq m - 1) \).

**Definition 2.** For any bMSC \( M = (P,E,\mathcal{V},C) \), an event sequence \( e_0 e_1 \ldots e_m \) is an untimed behaviour of \( M \) if and only if the following conditions hold:

• all events in \( E \) occur in the sequence, and each event occurs only once, i.e., \( \{e_0, e_1, \ldots, e_m\} = E \) and \( e_i \neq e_j \) for any \( i, j \) \( (i \neq j, 0 \leq i, j \leq m) \); and
• \( e_1, e_2, \ldots, e_m \) satisfy the visual order defined by \( \mathcal{V} \), i.e., for any \( e_i \) and \( e_j \), if \( (e_i, e_j) \in \mathcal{V} \), then \( 0 \leq i < j \leq m \).

For example, for the bMSC \( M_2 \) in Fig.1, the event sequence \( e_5 e_6 e_7 e_8 \) represents an untimed behaviour. It is not difficult to give an algorithm to check if there is an untimed behaviour for a given bMSC. Since we focus on timing analysis of MSCs in this paper, we assume that for any bMSC, there is at least one event sequence expressing its untimed behaviour.

We use timed event sequences to represent the behaviour of bMSCs. Any timed event sequence is of the form \( (e_0, t_0)^{\ldots} (e_i, t_i)^{\ldots} (e_m, t_m) \) where \( e_i \) is an event and \( t_i \) is a nonnegative real number for any \( i \) \( (0 \leq i \leq m) \), which describes that \( e_0 \) takes place \( t_0 \) time units after the system starts, then \( e_1 \) takes place \( t_1 \) time units after \( e_0 \) takes place, so on and so forth. At last \( e_m \) takes place \( t_m \) time units after \( e_{m-1} \) takes place. It follows that for any \( i \) \( (0 \leq i \leq m) \), the occurrence time of \( e_i \) is \( \sum_{j=0}^{i} t_j \).

**Definition 3.** A timed event sequence \( \sigma = (e_0, t_0)^{\ldots} (e_i, t_i)^{\ldots} (e_m, t_m) \) is a behaviour of a bMSC \( M = (P,E,\mathcal{V},C) \) if and only if the following conditions hold:

• \( e_0 e_1 \ldots e_m \) represents an untimed behaviour of \( M \), and
• \( t_0, t_1, \ldots, t_m \) satisfy the timing constraints described by \( C \), i.e., for any Boolean expression \( a \leq \sum_{i=0}^{n} c_i (f_i - f'_i) \leq b \) in \( C \), \( a \leq c_0 \delta_0 + c_1 \delta_1 + \cdots + c_n \delta_n \leq b \), where for each \( i \) \( (0 \leq i \leq n) \), if \( f_i = e_j \) and \( f'_i = e_k \), then
\[
\delta_i = \begin{cases} 
   t_{k+1} + t_{k+2} + \cdots + t_j & \text{if } j > k \\
   -(t_{j+1} + t_{j+2} + \cdots + t_k) & \text{if } j < k
\end{cases}
\]

Let \( \mathcal{L}(M) \) denote the set of the timed event sequences representing the behaviour of \( M \).

For example, for the bMSC \( M_2 \) in Fig.1, the timed event sequence

\[
(e_5, 3)^{\ldots} (e_6, 16)^{\ldots} (e_7, 10)^{\ldots} (e_8, 9)
\]

represents a behaviour, but the timed event sequence

\[
(e_5, 3)^{\ldots} (e_6, 14)^{\ldots} (e_7, 10)^{\ldots} (e_8, 9)
\]

does not represent a behaviour because it does not satisfy the timing constraint \( e_8 - e_5 \geq 34 \).

For describing timing constraints on the occurrence time of events, we introduce a special event \( \epsilon \) which represents the start of a system. For any bMSC \( M = (P,E,\mathcal{V},C) \) such that \( \epsilon \in E \), any timed event sequence in \( \mathcal{L}(M) \) is of the form

\[
(\epsilon, 0)^{\ldots} (e_1, t_1)^{\ldots} (e_2, t_2)^{\ldots} \ldots (e_m, t_m).
\]
2.2 MSC Specifications

An MSC specification is defined as a combination of a set of bMSCs and an hMSC describing their iterating and branching compositions. For describing the timing constraints enforced on two events in different bMSCs, a set of Boolean expressions of the form \( a \leq e - e' \leq b \) can be used as a complement.

Definition 4. An MSC specification (MSS) is a tuple \( S = (U, N, \text{succ}, \text{ref}, T) \)
- \( U \) is a finite set of bMSCs satisfying that for any \( M = (P, E, V, C) \in U \) and \( M' = (P', E', V', C') \in U \), if \( M \neq M' \), then \( E \cap E' = \emptyset \);
- \( N = \{ \top \} \cup I \cup \{ \perp \} \) is a finite set of nodes partitioned into three sets: singleton-set of start node, intermediate nodes, and singleton-set of end node, respectively;
- \( \text{succ} \subset N \times N \) is the relation which reflects the connectivity of the nodes in \( N \) such that any node in \( N \) is reachable from the start node and that the end node is reachable from any node in \( N \);
- \( \text{ref}: I \rightarrow U \) is a function that maps each intermediate node to a bMSC in \( U \); and
- \( T \) is a finite set of timing constraints of the form \( a \leq e - e' \leq b \) where \( e \) and \( e' \) occur in different bMSCs and \( 0 \leq a \leq b \) (\( b \) may be \( \infty \)).

For an MSS \( S = (U, N, \text{succ}, \text{ref}, T) \), a path segment is a sequence of intermediate nodes \( v_1^*v_2^*\ldots v_n^* \) satisfying that \( (v_{i-1}, v_i) \in \text{succ} \) for any \( i \) (\( 2 \leq i \leq n \)). A path segment is called simple if all its nodes are distinct. A path is a path segment \( v_1^*v_2^*\ldots v_n^* \) such that \( (v_1, v_1) \in \text{succ} \) and \( (v_n, v_n) \in \text{succ} \). A simple path is a path which is a simple path segment. For a simple path segment \( v_1^*v_2^*\ldots v_n^* \) such that \( (v_1, v_1) \in \text{succ} \), if there is \( v_1 \) \( (1 \leq i \leq n) \) such that \( (v_i, v_i) \in \text{succ} \), then the sequence \( v_1^*v_i^*v_{i+1}^*\ldots v_n^* \) is a loop and \( v_1 \) is a loop-start node.

To avoid the ambiguity in interpreting timing constraints related to loops, timing constraints are not allowed to combine event occurrences inside and outside of a loop, i.e., for any MSS \( S = (U, N, \text{succ}, \text{ref}, T) \), all timing constraints in \( T \) of the form \( a \leq e - e' \leq b \) must satisfy the following loop-closed condition:
- for any loop \( v_1^*v_2^*\ldots v_m^* \), if \( e \) occurs in \( \text{ref}(v_i) \) (\( 1 \leq i \leq m \)) and \( e' \) does not occur in any \( \text{ref}(v_j) \) (\( 1 \leq j < i \)), then there is no simple path segment \( v_1^*v_2^*\ldots v_n^* \) such that \( e' \) occurs in \( \text{ref}(v_n') \), \( e \) does not occur in any \( \text{ref}(v_k') \) (\( 1 \leq k \leq n \)), and that \( v_1' = v_1 \);
- for any loop \( v_1^*v_2^*\ldots v_m^* \), if \( e' \) occurs in \( \text{ref}(v_i) \) (\( 1 \leq i \leq m \)) and \( e \) does not occur in any \( \text{ref}(v_j) \) (\( i < j \leq m \)), then there is no simple path segment \( v_1^*v_2^*\ldots v_n^* \) such that \( v_1 = v_1' \), \( e \) occurs in \( \text{ref}(v_n') \), and that \( e' \) does not occur in any \( \text{ref}(v_k') \) (\( 1 \leq k \leq n \)); and
- for any loop \( v_1^*v_2^*\ldots v_m^* \), if \( e \) occurs in \( \text{ref}(v_i) \) and \( e' \) occurs in \( \text{ref}(v_j) \), then \( 1 \leq j < i \leq m \).

We interpret the timing constraints in MSSs by local semantics: select one path at a time and analyze its timing requirements, independently of other paths that may branch out of the selected one. We define the behaviour of an MSS \( S \) as the timed event sequences which are the concatenation of the timed event sequences representing the behaviour of the bMSCs which make up \( S \). We use \( \gamma \) to denote the concatenation of sequences.

Definition 5. For an MSS \( S = (U, N, \text{succ}, \text{ref}, T) \), a timed event sequence \( \sigma = (e_0, t_0)\gamma(e_1, t_1)\gamma\ldots \gamma(e_n, t_n) \) represents a behaviour of \( S \) if and only if the following condition holds:
- there is a path \( v_1^*v_2^*\ldots v_m^* \) such that \( \sigma = \sigma_1^*\sigma_2^*\ldots \sigma_m^* \), where \( \sigma_i \) is a behaviour of \( \text{ref}(v_i) \) for each \( i \) (\( 1 \leq i \leq m \)); and
- \( \sigma \) satisfies any timing constraint expressed by all Boolean expressions in \( T \), i.e., for any \( a \leq f - f' \leq b \) in \( T \), for any \( i, j \) (\( 0 \leq i < j \leq n \)) such that \( f^i = e_i \), \( f = e_j \), and that there is no \( k \) (\( i < k < j \)) satisfying \( f = e_k \lor f' = e_k \),

\[ a \leq t_{i+1} + t_{i+2} + \cdots + t_j \leq b. \]

Definition 6. For any MSS \( S = (U, N, \text{succ}, \text{ref}, T) \), for any path segment \( \rho = v_1^*v_2^*\ldots v_m^* \), let \( L(\rho) \) be the set of all timed event sequences which are of the form \( \sigma = (e_0, t_0)\gamma(e_1, t_1)\gamma\ldots \gamma(e_n, t_n) \) and satisfy that
- \( \sigma = \sigma_1^*\sigma_2^*\ldots \sigma_m^* \), where \( \sigma_i \) is a behaviour of \( \text{ref}(v_i) \) for each \( i \) (\( 1 \leq i \leq m \)); and
• \( \sigma \) satisfies all timing constraints expressed by all Boolean expressions in \( T \), i.e., for any \( a \leq f - f' \leq b \in T \), for any \( i, j \) (\( 0 \leq i < j \leq n \)) such that \( f' = e_i, f = e_j \), and that there is no \( k \) (\( i < k < j \)) satisfying \( f = e_k \lor f' = e_k \).

\[
a \leq t_{i+1} + t_{i+2} + \cdots + t_j \leq b.
\]

3 Checking MSC Specifications for Timing Inconsistency

In this section, we solve the verification problem of checking MSC specifications for timing inconsistency. For an MSS \( S \), a path \( \rho \) of \( S \) is timing consistent if and only if \( \mathcal{L}(\rho) \neq \emptyset \). An MSS \( S \) is timing inconsistent if and only if no path of \( S \) is timing consistent, i.e., every path \( \rho \) of \( S \) is such that \( \mathcal{L}(\rho) = \emptyset \). We know that for an MSS \( S \), the number of its paths could be infinite, and the length of a path of \( S \) could be infinite. So firstly we need to find a finite representation for the paths of an MSS, then solve the problem based on a finite set of finite paths.

3.1 Representing the Behaviour of MSC Specifications

It is well-known that the behaviour of automata can be expressed by regular expressions\(^8\). It follows that for any MSS, we can express the set of its paths by regular expressions. Here we represent the set of the path segments by normalised regular expressions (NREs), which are the rewritten forms of regular expressions. Let \( \varepsilon \) represent an empty sequence. For any path segment \( \rho, \varepsilon \rho = \rho \varepsilon = \rho \), and for any timed event sequence \( \sigma, \varepsilon \sigma = \sigma \varepsilon = \sigma \).

**Definition 7.** For an MSS \( S = (U, N, succ, ref, T) \), an NRE \( R \) and the set \( \mathcal{P}(R) \) of path segments are defined recursively as follows.

1. \( \varepsilon \) is an NRE, and \( \mathcal{P}(\varepsilon) = \{\varepsilon\} \).
2. If \( v \in U \), then \( v \) is an NRE, and \( \mathcal{P}(v) = \{v\} \).
3. If \( R_1 \) and \( R_2 \) are NREs, then \( R_1 \oplus R_2 \) is an NRE, and

\[
\mathcal{P}(R_1 \oplus R_2) = \mathcal{P}(R_1) \cup \mathcal{P}(R_2).
\]

4. If \( R \) is an NRE, then \( R^* \) is an NRE, and

\[
\mathcal{P}(R^*) = \{\rho_1 \hat{\rho}_2 \cdots \hat{\rho}_m \mid m \geq 0 \text{ and } \bigwedge_{i=1}^{m} (\rho_i \in \mathcal{P}(R))\},
\]

where \( \rho_1 \hat{\rho}_2 \cdots \hat{\rho}_m \hat{\varepsilon} \) when \( m = 0 \).

5. If \( R_1, R_2, \ldots, R_m \) are NREs such that either \( R_i = v \in U \) or \( R_i = R^* \) (\( R \) is an NRE) for any \( i \) (1 \( \leq i \leq m \)), then \( R_1 \hat{\rho}_2 \cdots \hat{\rho}_m \) is an NRE, and

\[
\mathcal{P}(R_1 \hat{\rho}_2 \cdots \hat{\rho}_m) = \{\rho_1 \hat{\rho}_2 \cdots \hat{\rho}_m \mid \rho_i \in \mathcal{P}(R_i) \text{ for any } i \ (1 \leq i \leq m)\}.
\]

There exist several well-developed polynomial-time algorithms to get the regular expression from a given automaton\(^9\). By the above definition, given an MSS \( S \), it is not difficult to get the NRE representing the set of the paths of \( S \), which can be constructed from a regular expression expressing the set of the paths of \( S \) by distributing \( \hat{\cdot} \) over \( \oplus \).

For an MSS \( S = (U, N, succ, ref, T) \), for an NRE \( R \), we denote the set of timed event sequences

\[
\{\sigma \mid \sigma \in \mathcal{L}(\rho), \rho \in \mathcal{P}(R)\}
\]

by \( \mathcal{L}(R) \ (\mathcal{L}(\varepsilon) = \{\varepsilon\}) \). It follows that for any MSS \( S \), the set of the timed event sequences representing the behaviour of \( S \) is \( \mathcal{L}(R) \) where \( R \) is an NRE representing the set of the paths of \( S \). In the next subsection, based on NREs, we will solve the problem of checking MSSs for timing inconsistency. In the following, we introduce some concepts concerning NREs that will be used in the next subsection.
For an NRE $R$, if there is an NRE $R_1$ occurring in $R$, then $R_1$ is a sub-expression of $R$.

For any timed event sequence $\sigma = (e_0, t_0)(e_1, t_1)\ldots(e_m, t_m)$, let $\tau(\sigma)$ denote the elapsing time on $\sigma$, i.e., $\tau(\sigma) = \sum_{i=0}^{m} t_i$. For any NRE $R$ such that $L(R) \neq \emptyset$, let $\theta(R)$ denote the infimum of the set $\{\tau(\sigma) : \sigma \in L(R)\}$.

A simple NRE is an NRE in which there is no occurrence of the combiners '$\ast$' (repetition) and $\oplus$ (union). It follows that any simple NRE is of the form $v_1 \ast v_2 \cdots v_n$ where any $v_i$ ($1 \leq i \leq n$) is a node.

For a simple NRE $R$ such that $L(R) \neq \emptyset$, if there is a timed event sequence in $L(R)$ of the form $(e_0, 0)\ast(e_1, 0)\cdots(e_m, 0)$, then $R$ is zero-simple, otherwise $R$ is non-zero-simple. In other words, a zero-simple NRE $R$ is such that there is a timed event sequence in $L(R)$ on which no time elapses, i.e., $\theta(R) = 0$. For an MSS $S = (U, N, succ, ref, T)$, it is clear that a simple NRE $R = v_1 \ast v_2 \cdots v_n$ is zero-simple if and only if for any $ref(v_i) = (P, E_i, V_i, C_i)$ ($1 \leq i \leq n$), any timing constraint in $C_i$ of the form $a \leq \sum_{j=0}^{m} c_j(e_j - e'_j) \leq b$ is such that $a \leq 0$ and $b \geq 0$, and any timing constraint in $T$ of the form $a \leq e - e' \leq b$ is such that $a = 0$.

By a normal form, we mean an NRE of the form $R_1 \oplus R_2 \oplus \ldots \oplus R_m$ where each $R_i$ ($1 \leq i \leq m$) is a simple NRE.

For any MSS $S = (U, N, succ, ref, T)$, since timing constraints are not allowed to combine event occurrences inside and outside of a loop, i.e., all timing constraints must satisfy the loop-closed condition given in Subsection 2.2, from now on we assume that any NRE $R$ satisfies the following star-closed condition:

- for any timing constraint in $T$ of the form $\alpha \leq e - e' \leq \beta$, there is no sub-expression $R'$ of $R$ satisfying
  $$ - R' = R_1 \ast R_2 \cdots \ast R_m \text{ where } R_1 = (v_1 \ast v_2 \cdots \ast v_n)^* \quad (v_i \in N \text{ for any } i (1 \leq i \leq n)) \text{ and } R_m = v \quad (v \in N), \text{ and}$$
  $$ - e' \text{ occurs in } ref(v_j) (1 \leq j \leq m) \text{ and does not occur in any } v_k (j < k \leq m), e \text{ occurs in } ref(v), \text{ and } e, e' \text{ do not occur in any } R_i (2 \leq i \leq m - 1);$$

- for any timing constraint in $T$ of the form $\alpha \leq e - e' \leq \beta$, there is no sub-expression $R'$ of $R$ satisfying
  $$ - R' = R_1 \ast R_2 \cdots \ast R_m \text{ where } R_1 = v \quad (v \in N) \text{ and } R_m = (v_1 \ast v_2 \cdots \ast v_n)^* \quad (v_i \in N \text{ for any } i (1 \leq i \leq n)), \text{ and}$$
  $$ - e' \text{ occurs in } ref(v), e \text{ occurs in } ref(v_j) (1 \leq j \leq m) \text{ and does not occur in any } v_k (1 \leq k < j), \text{ and } e, e' \text{ do not occur in any } R_i (2 \leq i \leq m - 1);$$

- for any timing constraint in $T$ of the form $\alpha \leq e - e' \leq \beta$, there is no sub-expression $R'$ of $R$ satisfying
  $$ - R' = (v_1 \ast v_2 \cdots \ast v_n)^* \quad (v_i \in N \text{ for any } i (1 \leq i \leq n)), \text{ and}$$
  $$ - \text{there are } i \text{ and } j \quad (1 \leq i < j \leq n) \text{ such that } e \text{ occurs in } ref(v_i) \text{ and } e' \text{ occurs in } ref(v_j) \text{ and } e, e' \text{ do not occur in } ref(v_k) (1 \leq k < i \lor j < k \leq n).$$

3.2 Model Checking Algorithm for Timing Inconsistency

It is clear that the problem of checking an MSC specification $S$ for timing inconsistency can be reduced to the problem of checking if $L(R) = \emptyset$ for the NRE $R$ which represents the set of the paths of $S$. In the following, we show how to check an NRE $R$ for the emptiness of $L(R)$.

We start from a simple case. We first consider checking simple NRE $R$ for the emptiness of $L(R)$. For an MSS $S = (U, N, succ, ref, T)$, suppose that the set of the paths of $S$ is represented by a simple NRE $R = v_1 \ast v_2 \cdots \ast v_n$ where $ref(v_i) = (P, E_i, V_i, C_i)$ for any $i (1 \leq i \leq n)$. Since there could be $E_i$ and $E_j$ ($1 \leq i, j \leq n, i \neq j$) such that $E_i = E_j$, by renaming, let $E_i \cap E_j = \emptyset$ for any $i, j (1 \leq i, j \leq n, i \neq j)$. Let $E = E_1 \cup E_2 \cup \ldots \cup E_n = \{e_1, e_2, \ldots, e_m\}$. Let $t_i$ represent the occurrence time of $e_i$ for any $i (1 \leq i \leq m)$. By Definitions 3 and 5, $t_1, t_2, \ldots, t_m$ must satisfy a group of linear constraints denoted as $lp(R)$. Hence, the problem of checking if $L(R) = \emptyset$ can be solved by checking if the group $lp(R)$ of linear constraints has no solution, which can be solved by linear programming.
For an MSS $S$, if the set of the paths of $S$ is represented by a normal form $R = R_1 \oplus R_2 \oplus \ldots \oplus R_m$, since each $R_i$ ($1 \leq i \leq m$) is a simple NRE, we can check if $S$ is timing inconsistent by checking if $\mathcal{L}(R_i) = \emptyset$ for each $i$ ($1 \leq i \leq m$).

For an MSS $S$ whose set of the paths is represented by a general NRE $R$, we attempt to solve the problem by finding a normal form $R'$ such that $\mathcal{L}(R) = \emptyset$ if and only if $\mathcal{L}(R') = \emptyset$. The process of finding $R'$ is a process of reducing the combinator $\ast$ from $R$. For describing how to remove the combinator $\ast$ formally, we need to introduce contexts.

Let $R$ be an NRE, and $R_1$ be a sub-expression of $R$. Replacing an occurrence of $R_1$ in $R$ with a letter $X$, we obtain a context of $X$, denoted as $C(X)$. Given an NRE $R$, any context $C(X)$ is associated with two real numbers $\varphi(C(X), R)$ and $\omega(C(X), R)$, which specify a lower bound and an upper bound of the constraints on the occurrence time enforced by the context on the variable $X$. If the context does not enforce any timing constraint on $X$ then

$$\varphi(C(X), R) = 0 \text{ and } \omega(C(X), R) = \infty.$$ 

**Definition 8.** For an MSS $S = (U, N, succ, ref, T)$, for an NRE $R$, a context $C(X)$ of $X$, and $\varphi(C(X), R)$ and $\omega(C(X), R)$ are defined recursively as follows.

1. $X$ is a context of $X$, $\varphi(X, R) = 0$ and $\omega(X, R) = \infty$ (no additional constraint).
2. If $C_1(X)$ is a context of $X$ and $R_1$ is an NRE, then $C(X) = C_1(X) \oplus R_1$ and $C(X) = R_1 \oplus C_1(X)$ are contexts of $X$, and $\varphi(C(X), R) = \varphi(C_1(X), R)$ and $\omega(C(X), R) = \omega(C_1(X), R)$ (no additional constraint).
3. If $C_1(X)$ is a context of $X$, then $C(X) = C_1(X)^\ast$ is a context of $X$, and $\varphi(C(X), R) = \varphi(C_1(X), R)$ and $\omega(C(X), R) = \omega(C_1(X), R)$ (no additional constraint).
4. If $C_1(X)$ is a context of $X$, $R_1 \cdots R_m$ is an NRE, then for any $k$ ($1 \leq k \leq m$), $C(X) = R_1 \cdots R_{k-1} C_1(X)^\ast R_{k+1} \cdots R_m$ is a context of $X$. $\varphi(C(X), R) = \max(\varphi(C_1(X), R), a)$ and $\omega(C(X), R) = \min(\omega(C_1(X), R), b)$ where $a$ is the maximal value of the set

$$\left\{ \begin{array}{l}
\{ a \leq e - e' \leq b \in T \} & \text{if } R_i \in ref(u), R_j = v, \\
\{ k < j \leq m \} & \text{and } e, e' \text{ do not occur in any } R_i (i < l < j) \text{ and } R_j
\end{array} \right\},$$

$b$ is the minimal value of the set

$$\left\{ \begin{array}{l}
\{ b \leq e - e' \leq b \in T \} & \text{if } R_i \in ref(u), R_j = v, \\
\{ k < j \leq m \} & \text{and } e, e' \text{ do not occur in any } R_i (i < l < j) \text{ and } R_j
\end{array} \right\},$$

and $a = 0$ and $b = \infty$ when the above set is empty (additional constraint enforced by timer events).

For any context $C(X)$, replacing $X$ in $C(X)$ with an NRE, say $R$, and distributing $\ast$ over $\oplus$, we obtain an NRE, denoted by $\mathcal{C}(R)$.

The following theorems show how to find a normal form $R'$ for a given NRE $R$ such that $\mathcal{L}(R) = \emptyset$ if and only if $\mathcal{L}(R') = \emptyset$ by removing the combinator $\ast$ from $R$. These theorems can be proved by induction on the structure of context. Their detailed proofs are omitted because of the space limitation.

For a real number $x$, let $\lfloor x \rfloor$ denote the floor of $x$. For an NRE $R$, let $R^j$ denote the $j$-repetition of $R$: $R^j = \overbrace{R \overline{\ldots R}^\ldots}^{j}(R^0 = \varepsilon)$.

**Theorem 1.** Let $R$ be an NRE and $R_1, R_2, \ldots, R_m$ be simple NREs. Let $C(X)$ be the context constructed by replacing an occurrence of $(R_1 \oplus R_2 \oplus \ldots \oplus R_m)^\ast$ with $X$ in $R$. Then $\mathcal{L}(C((R_1 \oplus R_2 \oplus \ldots \oplus R_m)^\ast)) = \emptyset$ if and only if $\mathcal{L}(C(R_1 \oplus R_2 \oplus \ldots \oplus R_m)) = \emptyset$.

**Theorem 2.** Let $R$ be an NRE and $R_1$ be a nonzero-simple NRE. Let $C(X)$ be the context constructed by replacing an occurrence of $R_1^\ast$ with $X$ in $R$. If $\omega(C(X), R_1) = \infty$, then $\mathcal{L}(C(R_1)) = \emptyset$ if and only if $\mathcal{L}(C(R_1^\ast)) = \emptyset$, where $p = \lfloor \varphi(C(X), R_1)/b \rfloor + 1$ ($0 < b \leq \theta(R_1)$).

**Theorem 3.** Let $R$ be an NRE and $R_1$ be a nonzero-simple NRE. Let $C(X)$ be the context constructed by replacing an occurrence of $R_1^\ast$ with $X$ in $R$. If $\omega(C(X), R_1) \neq \infty$, then $\mathcal{L}(C(R_1)) = \emptyset$ if and only if $\mathcal{L}(C(R_1^\ast)) = \emptyset$, where $p = \lfloor \omega(C(X), R_1)/b \rfloor + 1$ ($0 < b \leq \theta(R_1)$).
Let $\lambda$ be a bMSC $(P, E, O, C)$ such that $C = \emptyset$ and $E$ contains only one message $e$, i.e., $E = \{e\}$. Suppose that for any MSS $S = (U, N, succ, ref, T)$, $\lambda \in U$, $\Lambda \in N$ such that $ref(\Lambda) = \lambda$, and $e$ does not occur in any timing constraint in $T$.

**Theorem 4.** Let $R$ be an NRE and $R_1$ be a zero-simple NRE. Let $C(X)$ be the context constructed by replacing an occurrence of $R_1^T$ with $X$ in $R$. Then $\mathcal{L}(C(R_1^T)) = \emptyset$ if and only if $\mathcal{L}(C(\Lambda)) = \emptyset$.

Based on the above theorems, the algorithm, which checks an MSC specification $S$ for timing inconsistency, is now described as follows.

Step 0. Get NRE $R$ representing the set of the paths of $S$. Let $R' := R$.

Step 1. For $R'$, distributing $\bigcirc$ over $\oplus$, we obtain $Q$.

Step 2. Finding a sub-expression $Q_S$ of $Q$ which has one of the following forms:
1. $Q_S = (\oplus_{i=1}^{k} R_i)^*$, where each $R_i$ ($1 \leq i \leq k$) is a simple NRE.
2. $Q_S = R_1^T$, where $R_1$ is a non-zero simple NRE.
3. $Q_S = R_1^T$, where $R_1$ is a zero simple NRE.
4. $Q_S = R_1$, where $R_1$ is a simple NRE such that $\mathcal{L}(R_1) = \emptyset$.

If such $Q_S$ cannot be found, go to Step 7 (note that it is not difficult to prove that if we cannot find out such a $Q_S$, then $Q$ is a normal form); otherwise replacing the occurrence of $Q_S$ in $Q$ with $X$, we get a context $C_Q(X)$ such that $Q = C_Q(Q_S)$. Then, if $Q_S$ has the first form, go to Step 3; if $Q_S$ has the second form, go to Step 7; if $Q_S$ has the third form, go to Step 7; if $Q_S$ has the fourth form, go to Step 6.

Step 3. By Theorem 1, we transform $Q$ into $Q' = C_Q(R_1^T R_2^T \ldots R_m^T)$. Thus, let $R' := Q'$, and go to Step 1.

Step 4. If $\omega(C_Q(X), R_1) = \infty$, by Theorem 2, we transform $Q$ into $Q' = C_Q(\oplus_{i=0}^{p} R_1^T)$, where $p$ is defined in Theorem 2. Let $R' := Q'$, and go to Step 1. Otherwise, $\omega(C_Q(X), R_1) \neq \infty$, based on Theorem 3, we transform $Q$ into $Q' = C_Q(\bigcirc_{i=0}^{p} R_1^T)$, where $p$ is defined in Theorem 3. Let $R' := Q'$, and go to Step 1.

Step 5. By Theorem 4, we transform $Q$ into $Q' = C_Q(\Lambda)$. Let $R' := Q'$, and go to Step 1.

Step 6. Since $\mathcal{L}(R_1) = \emptyset$, $\mathcal{L}(R_1^T) = \{\varepsilon\}$. Thus, we transform $Q$ into $Q' = C_Q(\varepsilon)$. Let $R' := Q'$, and go to Step 1.

Step 7. Since $Q$ is a normal form now, we can check if $\mathcal{L}(Q) = \emptyset$. If $\mathcal{L}(Q) = \emptyset$, then $S$ is timing inconsistent; otherwise $S$ is not.

The above algorithm is based on linear programming. The linear programming problem has been well-studied, and can be solved with a polynomial-time algorithm in general. Indeed many software packages have been developed to efficiently find solutions for linear programs. In the algorithm, sometimes we need to unfold the combinator $\star$ (loop) a finite number of times (shown in Theorems 2 and 3). Each iteration will make the LP problem larger and hence this is the main source of complexity of the algorithm.

## 4 Conclusion

In this paper, we have considered the problem of checking MSC specifications for timing inconsistency. We have given an algorithm to solve the problem that is based on linear programming. In [6], MSC specifications are interpreted as a global state automaton. Theoretically the problem can thus be solved by checking a timed automaton for emptiness, which is of high complexity. Our approach is based directly on MSC specifications and so we avoid the generation of the state space altogether and also the involved complexity.

An MSC specification is timing inconsistent means that no execution scenario described by the MSC specification is timing consistent. It is allowed that some scenarios identified are never executed in a specification or design. But it is really an error that in a specification or design, no scenario can be executed according to the timing constraints enforced by the system. So it is necessary to check MSC specifications for timing inconsistency. We have implemented the proposed algorithm. An important topic for the future work is doing case studies in practical use.
References


LI Xuandong is a professor in the Department of Computer Science, Nanjing University. He received his Ph.D. degree in computer science from Nanjing University in 1994. His research interests include formal methods and object-oriented technology.

TAN Wenkai is a graduate student in the Department of Computer Science, Nanjing University. He received his B.S. degree in computer science from Nanjing University in 1998. His research interests include verification of real-time systems, model checking.

ZHENG Guoliang is a professor in the Department of Computer Science, Nanjing University. He received his B.S. degree in computer science from Nanjing University in 1961. His main research area is software engineering.