Matching Scenarios with Timing Constraints*

Prakash Chandrasekaran and Madhavan Mukund

Chennai Mathematical Institute, Chennai, India
{prakash, madhavan}@cmi.ac.in

Abstract. Networks of communicating finite-state machines equipped with local clocks generate timed MSCs. We consider the problem of checking whether these timed MSCs are "consistent" with those provided in a timed MSC specification. In general, the specification may be both positive and negative. The system should execute all positive scenarios "sensibly". On the other hand, negative scenarios rule out some behaviours as illegal. This is more complicated than the corresponding problem in the untimed case because even a single timed MSC specification implicitly describes an infinite family of timed scenarios. We outline an approach to solve this problem that can be automated using UPPAAL.

1 Introduction

In a distributed system, several agents interact with each other to generate a global behaviour. The interaction between these agents is usually described in terms of scenarios, using mechanisms such as use-cases and message sequence charts (MSCs) [8].

In general, scenarios could be of two types, positive and negative. Positive scenarios are those that the system is designed to execute—for instance, these may describe a handshaking protocol to set up a reliable communication channel between two hosts on a network. Negative scenarios indicate undesirable behaviours, such as a situation when both hosts independently initiate the activity of setting up a channel, leading to a collision.

This leads to a natural verification problem: given a distributed system and a scenario, does the system exhibit the scenario? In the context of message sequence charts, this is referred to as the scenario matching problem, for which efficient algorithms have been identified [10]. An approach to solve this problem using the modelchecker SPIN was proposed in [4].

In this paper, we extend the study of scenario matching to timed systems. We consider communicating finite-state machines equipped with local clocks. Clock constraints are used to guard transitions and specify location invariants, as in other models of timed automata [3]. Just as the runs of timed automata can be described in terms of timed words, the interactions exhibited by communicating finite-state machines with clocks can be described using timed MSCs.

* Partially supported by *Timed-DISCOVERI*, a project under the Indo-French Networking Programme.

E. Asarin and P. Bouyer (Eds.): FORMATS 2006, LNCS 4202, pp. 98–112, 2006.

We define a version of scenarios with timing constraints that we call timed MSC templates. These templates are built from fixed underlying MSCs by associating a lower and upper bound on the time interval between certain pairs of events. Timed MSC templates are a natural and useful extension of the untimed notation for scenarios, because protocol specifications typically include timing requirements for message exchanges, as well as descriptions of how to recover from timeouts.

In general, a timed MSC template is compatible with infinitely many timed MSCs. Thus, the scenario matching problem is already more complicated than in the untimed case, where a single scenario describes exactly one pattern of interaction. In our setting, the scenario matching problem can be reformulated in terms of checking whether the intersection of two collections of timed MSCs is nonempty.

We propose an approach to tackle this problem using the modelchecking tool UPPAAL, which is designed to verify properties of timed systems. Unfortunately, the basic system model of UPPAAL consists of a network of timed automata that communicate via synchronous handshakes, rather than message-passing. We thus need to code up message-passing channels by creating special processes to model buffers. However, we can exploit the handshake mechanism to synchronize the system with the template to be verified. This automatically reduces the behaviours of the system to those that are consistent with the template. The scenario matching problem can then be easily transformed into a modelchecking question for UPPAAL to verify on the composite system.

The paper is organized as follows. In the next two sections, we formally define timed MSCs and timed message-passing automata. This enables us to precisely define the scenario matching problem for timed systems in Section 4. In Section 5, we describe our approach to address the scenario matching problem in UPPAAL. Next, we look at issues related to the expressiveness of our timed message-passing automaton model. In Section 7, we examine some optimizations that can be introduced when translating the scenario matching problem to UPPAAL, to improve the efficiency of verification. We conclude with a brief discussion.

2 Timed MSCs

2.1 Message Sequence Charts

Let $\mathcal{P} = \{p, q, r, \ldots\}$ be a finite set of processes (agents) that communicate with each other through messages via reliable FIFO channels using a finite set of message types $\mathcal{M}$. For $p \in \mathcal{P}$, let $\Sigma_p = \{p!q(m), p?q(m) \mid p \neq q \in \mathcal{P}, m \in \mathcal{M}\}$ be the set of communication actions in which p participates. The action $p!q(m)$ is read as *p sends the message m to q* and the action $p?q(m)$ is read as *p receives the message m from q*. We set $\Sigma = \bigcup_{p \in \mathcal{P}} \Sigma_p$. We also denote the set of *channels* by $Ch = \{(p, q) \mid p \neq q\}$.

Labelled posets. A Σ-labelled poset is a structure $M = (E, \leq, \lambda)$ where $(E, \leq)$ is a poset and $\lambda : E \to \Sigma$ is a labelling function. For $e \in E$, let $\downarrow e = \{e' \mid e' \leq e\}$.