Zone-Based Synthesis
of Strict 2-Phase Fault Recovery

Fathiyeh Faghih and Borzoo Bonakdarpour
School of Computer Science, University of Waterloo, Canada

Abstract. In this paper, we focus on efficient synthesis of fault-tolerant timed models from their fault-intolerant version. We propose an algorithm that takes a timed automaton, a set of fault actions, and a set of safety and bounded-time response properties as input, and utilizes a space-efficient symbolic representation of the timed automaton (called the zone graph) to synthesize a fault-tolerant timed automaton as output. The output automaton satisfies strict phased recovery, where it is guaranteed that the output model behaves similarly to the input model in the absence of faults and in the presence of faults, fault recovery is achieved in two phases, each satisfying certain safety and timing constraints.

1 Introduction

Dependability and time-predictability are two vital properties of most embedded (especially, safety/mission-critical) systems. Consequently, providing fault-tolerance and meeting timing constraints are two inevitable aspects of dependable real-time embedded systems. However, these two features have conflicting natures; i.e., fault-tolerance deals with unanticipated faults, while meeting timing constraints requires time predictability. This conflict inevitably makes design and analysis of fault-tolerant real-time systems a tedious and error-prone task.

Let $Q$ and $P$ be two predicates that should be reached in phase 1 and 2 of recovery within different time bounds, respectively. In [2], the authors have shown that if $Q$ is not required to be closed in the execution of recovery transitions, then synthesizing a timed automaton [1] with 2-phase recovery is NP-complete in the size of the detailed region graph [1] of the input automaton. On the contrary, if the closure of $Q$ is required and, moreover, $P \subseteq Q$, then the synthesis problem can be solved in polynomial time. The polynomial-time algorithm presented in [2] to solve the latter problem is only an evidence for proving the complexity of the problem and is not an efficient practical solution with potential for implementation. This is simply because the size of a detailed region graph grows incredibly huge even for small models.

With this motivation, in this paper, we propose a time- and space-efficient algorithm for synthesizing timed automata that provide 2-phase recovery, where $Q$ is required to be closed and $P \subseteq Q$, while no new behaviors are added in the absence of faults.
2 Problem Statement

Given are a fault-intolerant timed automaton $TAD$ with semantic model $\mathcal{SM} = (S, s_0, T)$ and legitimates states $LS$, a set $F$ of faults, and specification $SPEC$, such that $TAD$ satisfies $SPEC$ by starting from any legitimate state $LS$ (denoted $TAD \models_{LS} SPEC$). Our goal is to develop an algorithm for synthesizing an automaton $TAD'$ with legitimate states $LS'$ from $TAD$, such that $TAD'$ is $F$-tolerant to $SPEC$ from $LS'$ \cite{2}. By $F$-tolerant, we mean that when the state of a system is perturbed by faults, the system is required to either directly return to its legitimate states $LS$ within $\theta \in \mathbb{Z}_{\geq 0}$ time units, or, if direct recovery is not feasible, then it should first reach an intermediate recovery predicate $Q$ within $\theta \in \mathbb{Z}_{\geq 0}$ (i.e., phase 1), from where the system reaches $LS'$ within $\delta \in \mathbb{Z}_{\geq 0}$ time units (i.e., phase 2). We require that the algorithm for adding fault tolerance does not introduce new behaviors to $TAD$ in the absence of faults. These constraints are formally stated below, where $T \mid LS$ denotes the set of transitions $T$ that start and end in $LS$.

| Problem statement. Given a fault-intolerant timed automaton $TAD$ with semantic model $\mathcal{SM} = (S, s_0, T)$, a set $F$ of faults, intermediate predicate $Q$, where $LS \subseteq Q$, and specification $SPEC$, such that $TAD \models_{LS} SPEC$, our goal is to propose an algorithm for synthesizing an automaton $TAD'$ with $\mathcal{SM}' = (S', s'_0, T')$, and legitimate states $LS'$ from $TAD$, such that:

1. $LS' \subseteq LS$,
2. $Q$ is closed in $T'$,
3. $(T' \mid LS') \subseteq (T \mid LS')$, and
4. $TAD'$ is $F$-tolerant to $SPEC$ from $LS'$.

3 The Synthesis Algorithm

Our zone-based algorithm consists of the following steps:

1. (Automaton enhancement) The input model is enhanced, so that the corresponding zone graph is more efficient and is augmented with delay transitions that can be utilized for adding 2-phase recovery.
2. (Zone graph generation) Next, the zone graph (a space-efficient representation of a timed automata) of the enhanced input automaton is generated. We utilize an existing algorithm from the literature of verification for this step.
3. (Adding recovery behavior) To enable 2-phase recovery, we add possible transitions among the zones of the zone graph. In this step, new zones may be added to the zone graph.
4. (Backward zone generation) For the newly added zones in the last step, we identify the backward reachable zones to ensure that the new zones do not introduce terminating computations.