Conflict-Tolerant Real-time Specifications in Metric Temporal Logic

Sumesh Divakaran Deepak D'Souza
Raj Mohan M.
Department of Computer Science and Automation,
Indian Institute of Science,
Bangalore 560012, India.

October 16, 2009

Abstract

A framework based on the notion of "conflict-tolerance" was proposed in [3, 4] as a compositional methodology for developing and reasoning about systems that are composed of multiple independent controllers. A central notion in this framework is that of a "conflict-tolerant" specification for a controller. In this work we propose a way of specifying conflict-tolerant real-time controllers in Metric Interval Temporal Logic (MITL). We call our logic CT-MITL for Conflict-Tolerant MITL. We consider the associated verification and synthesis problems for CT-MITL and give decision procedures to solve them.

1 Introduction

A framework based on the notion of "conflict-tolerance" was proposed in [3, 4] as a way of developing and reasoning about systems that are composed of a base system along with multiple independent controllers that each implement a certain feature for the system. Such systems appear commonly in software intensive domains, examples of which include a telecom switch which provides different features to subscribers, like call forwarding and call screening; or an automobile with several time-dependent features like cruise control and stability control. Typically the controller for each feature is developed independently, and the controllers are all integrated together using a hand-coded supervisory controller. Unfortunately in certain configurations of the system – as the reader may well imagine for the example features mentioned above – the individual controllers may profer conflicting advises on how the system should proceed next. These conflicts are typically resolved by suspending the lower-priority controller and then waiting for a "reset" state of the system before restarting the controller. As a result the system loses out on the suspended feature's utility during this period.

The framework in [3, 4] proposes a way of designing each controller so that, based on a priority ordering among the features, it is easy to compose them in a way in

which each controller is utilised "maximally." Thus each controller's advice is taken at all times except when *each* of its advised actions is in conflict with a higher priority controller. The key idea in this framework is to specify a "conflict-tolerant" behaviour for each feature, and then to build a controller for each feature that meets its conflict-tolerant specification. Unlike a classical safety specification, which can be viewed as a prefix-closed language of behaviours, a conflict-tolerant specification is an *advice function* which specifies a safety language for *each* possible finite behaviour of the system. This is depicted in Fig. 1: If one considers the set of all possible behaviours of the system as a tree growing downwards, then part (a) shows the shaded "cone" denoting a classical safety language, and part (b) depicts what a tolerant specification may look like, with safety cones prescribed for each possible behaviour of the system. A controller for such a specification must itself be "tolerant" in that it not only advises the system on what actions to take next (like a classical controller), but also keeps track of possible deviations from its advice, and goes on to advise next events so as to control the *subsequent* behaviour of the system. A conflict-tolerant controller satisfies a tolerant specification given as an advice function f if after *every* system behaviour σ, the subsequent controlled behaviour of the system stays within the safety language $f(\sigma)$.

Figure 1: (a): A classical safety specification and (b) a conflict-tolerant specification.

An important component missing in the framework of [3, 4] is the ability to specify conflict-tolerant specifications in a popular specification language like temporal logic. In this paper our aim is to fill this gap in the domain of real-time systems, by proposing a way of specifying conflict-tolerant specifications in Metric Interval Temporal Logic (MITL) [6, 2].

The logic we propose, called CT-MITL for "Conflict-Tolerant MITL," is a syntactic fragment of MITL. A CT-MITL specification is a conjunction of formulas of the form $\Box(\varphi \implies \psi)$, where φ is a past-MITL formula, and ψ is a disjunction of formulas of the form c, where c is a system action or the symbolic event δ which stands for "time elapse". A CT-MITL formula defines an "immediate" advice function in a natural way: at the end of any behaviour σ, we check whether the past formula φ is true, and if so, advise a set of next actions that satisfy ψ.

The associated verification problem for CT-MITL is to check, given a base system $\mathcal{B}$ and a conflict-tolerant controller C (both modelled as Alur-Dill timed transition systems), and a conflict-tolerant specification in the form of a CT-MITL formula θ, whether C satisfies the *advice function* induced by θ, with respect to the given base system $\mathcal{B}$ (as described above). We note that an advice function is in general a richer object than a classical safety specification, and thus the verification problem for CT-MITL is more general different than the classical verification problem for MITL. In general, a

controller C may satisfy θ as a classical MITL specification, but *may not* satisfy it as a conflict-tolerant specification.

Nevertheless, we show that the verification problem, as well as the associated feasibility and synthesis problems, for CT-MITL can be solved algorithmically, using essentially the same technique as for classical MITL [2, 7, 8]. The main step is to build for a given past-MITL formula φ a deterministic transition system that "monitors" the truth of φ along every timed word it reads. The construction we give is inspired by the simple *compositional* construction of a deterministic timed automaton for a past-MITL formula given in [8]. However our technique is more general as it handles the full fragment of past-MITL and not just closed intervals as done in [8]. Further our construction for the main inductive step of $\diamondsuit_I\varphi$, which we term "delay-then-extend," is provably clock-optimal. In fact, we show that this step can be implemented very simply using just two off-the-shelf "delay" and "extend" components, available in tools like Simulink.

A more detailed technical report is available in [10]. Our logic and techniques can also be specialized easily to the discrete (untimed) setting as detailed in [11].

2 Preliminaries

For an alphabet of symbols A we denote the set of words over A by A^*, and by ϵ the empty word. Let $\mathbb{R}_{\geq 0}$ and $\mathbb{Q}_{\geq 0}$ denote the set of non-negative reals and rationals respectively. We will use the standard notation to describe intervals of reals. So for example the interval $(1, 2]$ denotes the set $\{t \in \mathbb{R}_{\geq 0} \mid 1 < t \leq 2\}$. An interval is *non-singular* if the set it denotes is not a singleton set. Whenever convenient we use '$\langle$' (resp. '$\rangle$') to denote a left-open or left-closed (resp. right-open or right-closed) interval bracket.

A *timed word* σ over Σ is a string in $(\Sigma \cup \mathbb{R}_{\geq 0})^*$ of the form $d_0a_1d_1a_2d_2 \cdots a_nd_n$, where $n \geq 0$, $d_0, d_n \in \mathbb{R}_{\geq 0}$, $d_i \in \mathbb{R}_{>0}$ for $0 < i < n$, and $a_j \in \Sigma$ for $1 \leq j \leq n$. We use the notation $dur(\sigma)$ to denote the duration of σ, in this case $\sum_{i=0}^{n} d_i$. We use the symbol ε to denote the empty timed word whose representation is 0.

Let $\sigma = d_0a_1d_1 \cdots a_md_m$, and $\tau = e_0b_1e_1 \cdots b_ne_n$, be timed words such that it is not the case that: $n, m > 0$ and $d_m = e_n = 0$. Then we define the *concatenation* of σ and τ, denoted $\sigma \cdot \tau$ (or simply $\sigma\tau$ when clear from the context), to be the timed word $d_0a_1d_1 \cdots a_m(d_m + e_0)b_1e_1 \cdots b_ne_n$. We say a timed word σ is a *prefix* of a timed word τ, written $\sigma \preceq \tau$, if there exists a timed word σ' such that $\sigma \cdot \sigma' = \tau$. We denote the set of all timed words over Σ by $T\Sigma^*$. We note that our timed words are essentially Alur-Dill timed words except that we use a delay-based representation and allow timed words to end with delays.

It will often be convenient to view a timed word σ over Σ as a map from $[0, dur(\sigma)]$ to $\Sigma \cup \{\delta\}$, which tells us whether the event at time t in $[0, dur(\sigma)]$ is an action point $a \in \Sigma$, or the symbolic event "δ" which denotes a non-action point or "time elapse." Thus if $\sigma = d_0a_1d_1 \cdots a_nd_n$, for each $t \in [0, dur(\sigma)]$ we can define

$$\sigma(t) = \begin{cases} a_k & \text{if } n > 0 \text{ and } \exists k \in \{1, \ldots, n\} : \ t = \Sigma_{i=0}^{k-1} d_i \\ \delta & \text{otherwise.} \end{cases}$$

By $last(\sigma)$ we will mean the value $\sigma(dur(\sigma))$ in $\Sigma \cup \{\delta\}$. For $t \in [0, dur(\sigma)]$ we use the notation $\sigma[0,t)$ to denote the prefix of σ of duration t which does not end in an action point. Thus $\sigma[0,t) = \tau$ where $\tau \preceq \sigma$, $dur(\tau) = t$, and $last(\tau) = \delta$.

A *timed language* over Σ is simply a set of timed words over Σ. A timed language L is called *prefix-closed* if whenever $\sigma \in L$ and $\tau \preceq \sigma$, we have $\tau \in L$. We denote by $L_{\preceq}$ the prefix-closure of L. For a timed language $L \subseteq T\Sigma^*$ and a timed word σ, we denote by $ext_\sigma(L)$, the *set of extensions* of σ that are in L. Thus $ext_\sigma(L) = \{\tau \in T\Sigma^* \mid \sigma \cdot \tau \in L\}$.

We use a variant of Alur-Dill timed transition systems [1] which have "time can progress" conditions [9] as state invariants, to model the systems we consider. To begin with let C be a finite set of clocks. A *valuation* for the clocks in C is a map $v : C \to \mathbb{R}_{\geq 0}$. We denote by $\vec{0}$ the valuation which maps all clocks in C to 0. For a valuation v and $t \in \mathbb{R}_{\geq 0}$, by $v + t$ we mean the valuation that maps each $x \in C$ to $v(x) + t$, and by $v[0/X]$, for a subset of clocks X of C, the valuation which maps each x in X to 0, and each x in $C - X$ to $v(x)$. A *clock constraint* g over C is a boolean combination of atomic constraints of the form $x \sim c$, where x is a clock in C, $\sim \in \{<, \leq, =, >, \geq\}$ and c is a rational constant. We write $v \models g$ to say that the valuation v satisfies the clock constraint g, with the expected meaning. By $\Phi(C)$ we denote the set of clock constraints over C.

A *timed transition system* (TTS) over an alphabet Σ is a structure of the form $\mathcal{T} = (Q, C, s, \to, tcp)$, where Q is a finite set of states, C is a finite set of clocks, $s \in Q$ is the initial state, $\to \subseteq Q \times \Sigma \cup \{\epsilon\} \times \Phi(C) \times 2^C \times Q$ is a finite set of transitions, and $tcp : Q \to \Phi(C)$ specifies the condition under which *time can progress* in a given state. A *configuration* of $\mathcal{T}$ is a pair (q, v) where $q \in Q$ and v is a valuation for the clocks in C. From a given configuration (q, v) of $\mathcal{T}$, there are two kinds of transitions:

discrete: $(q, v) \xrightarrow{c} (q', v')$ where $c \in \Sigma \cup \{\epsilon\}$, if $(q, c, g, X, q') \in \to$, $v \models g$, and $v' = v[0/X]$.

delay: $(q, v) \xrightarrow{d} (q, v + d)$ where $d \in \mathbb{R}_{\geq 0}$, if for all $0 \leq d' < d$, $(v + d') \models tcp(q)$.

A *run* of $\mathcal{T}$ on a timed word σ is a sequence of time points

$$0 = t_0 \leq t_1 < t_2 < \cdots < t_n \leq t_{n+1} = dur(\sigma),$$

with $n \geq 0$, along with a sequence of configurations of $\mathcal{T}$, $(q_0, v_0), (q_1, v_1), \ldots, (q_{2n+1}, v_{2n+1})$, satisfying

- $(q_0, v_0) = (q, v)$
- for each $i \in \{1, \ldots, n+1\}$, $(q_{2i-2}, v_{2i-2}) \xrightarrow{t_i - t_{i-1}} (q_{2i-1}, v_{2i-1})$
- for each $i \in \{1, \ldots, n\}$, $(q_{2i-1}, v_{2i-1}) \xrightarrow{c_i} (q_{2i}, v_{2i})$ where $c_i = \sigma(t_i)$ if $\sigma(t_i) \in \Sigma$, else ϵ.

The timed language of $\mathcal{T}$, denoted $L(\mathcal{T})$, is the set of all timed words on which $\mathcal{T}$ has a run starting from the initial configuration $(s, \vec{0})$. Similarly, we denote by $L_{(q,v)}(\mathcal{T})$ the language of timed words on which $\mathcal{T}$ has a run starting from the configuration (q, v).

We say a TTS $\mathcal{T}$ is *deterministic* if there is at most one run of $\mathcal{T}$ on any timed word $\sigma \in T\Sigma^*$. A sufficient condition for $\mathcal{T}$ to be deterministic is that for every pair of

transitions (q, c, g_1, X_1, q') and (q, c, g_2, X_2, q''), the constraint $g_1 \wedge g_2$ is not satisfiable. Similarly, for every state $q \in Q$, and every transition (q, ϵ, g, X, q'), $g \wedge tcp(q)$ should not be satisfiable. A TTS $\mathcal{T}$ is *complete* if there is at least one run of $\mathcal{T}$ on any timed word in $T\Sigma^*$. For a deterministic and complete TTS $\mathcal{T}$, there is a unique run of $\mathcal{T}$ on a given timed word σ. Let (q, v) be the unique configuration reached by $\mathcal{T}$ on σ and let us denote it by $config_{\mathcal{T}}(\sigma)$. Then we define $L_\sigma(\mathcal{T}) = L_{config_{\mathcal{T}}(\sigma)}(\mathcal{T})$.

It will be convenient to sometimes use constraints over both the states and clocks in a TTS. A *configuration constraint* g over a set of states Q and a set of clocks C is a boolean combination of atomic constraints of the form q (for $q \in Q$) and $x \sim c$ (as for clock constraints). Once again, the notion of when a configuration (q, v) satisfies a configuration constraint g is defined in the expected way. A TTS with configuration constraints is a TTS which uses configuration constraints instead of clock constraints, and the notions of runs, etc, are defined in a similar manner as for TTS's.

Let $\mathcal{T}_1 = (Q_1, C_1, s_1, \rightarrow_1, tcp_1)$ and $\mathcal{T}_2 = (Q_2, C_2, s_2, \rightarrow_2, tcp_2)$ be two timed transition systems, possibly with configuration constraints, over the same alphabet Σ. We assume that the set of clocks C_1 and C_2, and states Q_1 and Q_2 are disjoint. Then the *synchronized product* of $\mathcal{T}_1$ and $\mathcal{T}_2$, denoted by $\mathcal{T}_1 \parallel \mathcal{T}_2$, is given by the TTS $(Q_1 \times Q_2, C_1 \cup C_2, (s_1, s_2), \rightarrow, tcp)$ where $\rightarrow$ and tcp are defined as follows.

- $((p, q), g, c, X, (p', q')) \in \rightarrow$ if either there exist transitions $(p, g_1, c, X_1, p') \in \rightarrow_1$ and $(q, g_2, c, X_2, q') \in \rightarrow_2$, and $g = g_1 \wedge g_2$, and $X = X_1 \cup X_2$; or $c = \epsilon$ and there exists a transition $(p, g, \epsilon, X, p') \in \rightarrow_1$, and $q = q'$ (or vice-versa).
- $tcp((p, q)) = tcp_1(p) \wedge tcp_2(q)$.

We note that the synchronized products as defined above generates the intersection of the timed languages $L(\mathcal{T}_1)$ and $L(\mathcal{T}_2)$.

Finally, we define the notion of an "open" TTS which we will make use of in our inductive construction of monitoring TTS's. An *open* TTS (over Σ) with respect to the pairwise-disjoint set of states and clocks $Q_1, C_1, \ldots, Q_n, C_n$, is a structure $\mathcal{U} = (Q, C, s, \rightarrow, tcp)$, similar to a TTS over Σ, that uses "open" configuration constraints over $Q_1, C_1, \ldots Q_n, C_n$. *Open* configuration constraints are boolean combinations of configuration constraints over Q, C, and atomic constraints of the form $\mathcal{T}_i.q$ (where $q \in Q_i$), and $\mathcal{T}_i.x \sim c$ (where $x \in C_i$). Given TTS's $\mathcal{T}_i = (Q_i, C_i, s_i, \rightarrow_i, tcp_i)$, we can define the composition of $\mathcal{U}$ and $\mathcal{T}_1, \ldots, \mathcal{T}_n$, denoted $\mathcal{U} \parallel \mathcal{T}_1 \parallel \cdots \parallel \mathcal{T}_n$, to be the TTS with configuration constraints obtained by taking the parallel composition of $\mathcal{U}, \mathcal{T}_1, \ldots, \mathcal{T}_n$ as defined above, but where the constraints $\mathcal{T}_i.q$ and $\mathcal{T}_i.y \sim c$ are interpreted as the configuration constraints $\bigvee_{u \in Q \times Q_1 \times \cdots \times Q_n,\ u(i)=q} u$ and $y \sim c$ respectively.

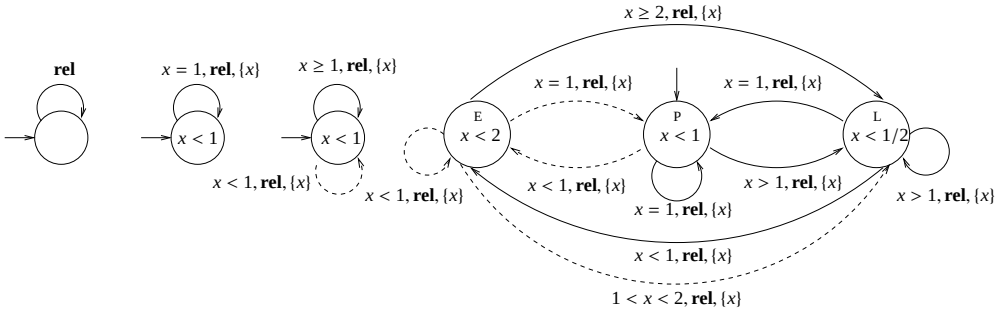
3 Conflict-tolerant Controllers

In this section we recall some of the key notions in the real-time conflict-tolerant framework from [4]. We begin with the notion of a base system. Some of the example systems we use here are adapted from [4, 5].

A base system is modelled as a TTS over an alphabet comprising "system" or "controllable" events and "environment" or uncontrollable events. We call such an alphabet

a *partitioned* alphabet, and it is of the form (Σ_e, Σ_s). We will use the convention that $\Sigma = \Sigma_e \cup \Sigma_s$. Let us fix a partitioned alphabet (Σ_e, Σ_s) for the rest of this section.

Definition 1. *A* base system *(or* plant*) over* (Σ_e, Σ_s) *is a deterministic timed transition system* $\mathcal{B}$ *over* Σ*, which we assume to be* non-blocking *in that whenever* $\sigma \in L(\mathcal{B})$*, then* $ext_\sigma(L(\mathcal{B})) \neq \{\varepsilon\}$.



(a) (b) (c) (d)

Figure 2: (a) A base system, (b) a classical controller, and (c), (d) two conflict-tolerant controllers.

Fig. 2 (a) shows a simple base system model which has only one system event "**rel**" (say for "releasing" a unit of lubricant), and no environment events.

In the classical framework, a real-time safety specification for a controller is given by a prefix-closed timed language. A controller for a given base system satisfies this specification if all behaviours of the controlled base system are contained in the specified safety language. For example, the TTS shown in Fig. 2(b) is a classical controller for the example base system that ensures a **rel** event after every 1 time unit.

A *conflict-tolerant specification* on the other hand is a collection of safety languages, one for *each* possible behaviour of the base system. This is formalised as an "advice function" below, which advises a timed language of future extensions for each possible behaviour.

Definition 2. *A (real-time)* advice function *over an alphabet* Σ *is a mapping* $f : T\Sigma^* \to 2^{T\Sigma^*}$ *which satisfies the following conditions:*

- *for every timed word* σ *over* Σ*,* $f(\sigma)$ *is a prefix-closed language.*
- f *is consistent, i.e. for all timed words* σ, τ *over* Σ*, if* $\tau \in f(\sigma)$ *then* $f(\sigma\tau) = ext_\tau(f(\sigma))$.

An alternate way of describing an advice function is an *immediate* advice function:

Definition 3. *An* immediate advice function *over* Σ *is a mapping* $g : T\Sigma^* \to 2^{\Sigma \cup \{\delta\}}$.

Given a timed word σ over Σ, and a timepoint $t \in [0, dur(\sigma)]$, we say σ is *according to* the immediate advice g at time t if $\sigma(t) \in g(\sigma[0,t))$. The immediate advice function g above, induces an advice function $f_g : T\Sigma^* \to 2^{T\Sigma^*}$ given by:

$$f_g(\sigma) = \{\, \tau \in T\Sigma^* \mid \sigma\tau \text{ is according to } g \text{ in } \langle dur(\sigma), dur(\sigma\tau)]\}$$

where '$\langle$' = '[' if $\sigma(t) = \delta$ and '(' otherwise. It is easy to verify that the two conditions in Def. 2 are satisfied by f_g, and hence that it is a valid advice function.

A conflict-tolerant controller is similar to a classical controller which synchronizes with the base system and controls the choice of possible next system events available to the base system. The main difference however is that a conflict-tolerant controller also keeps track of the system events that are *against* its advice, and goes on to control the *subsequent* behaviour of the system. A real-time conflict-tolerant controller is modelled as an annotated TTS described below.

A *conflict-tolerant* timed transition system (CTTS) over Σ is a tuple $\mathcal{T}' = (\mathcal{T}, N, tcp')$ where $\mathcal{T} = (Q, C, s, \rightarrow, tcp)$ is a deterministic TTS over Σ, $N \subseteq \rightarrow$ is a subset of transitions designated as *not-advised*, and tcp' is the advised time-can-progress condition for states of $\mathcal{T}$. Let (q, v) be a configuration of the TTS $\mathcal{T}$. Then the *unconstrained* language generated by $\mathcal{T}'$ starting from (q, v), denoted $L_{(q,v)}(\mathcal{T}')$, is defined to be $L_{(q,v)}(\mathcal{T})$. The *constrained* language generated by $\mathcal{T}'$ starting from (q, v) is denoted $L^c_{(q,v)}(\mathcal{T}')$, and defined to be $L_{(q,v)}(\widetilde{\mathcal{T}})$, where $\widetilde{\mathcal{T}}$ is the transition system obtained from $\mathcal{T}$ by deleting all not-advised transitions (i.e. transitions in N) and replacing the time-can-progress condition tcp by tcp'. Let σ be a timed word in $L(\mathcal{T})$. Let (q, v) be the unique configuration reached by $\mathcal{T}$ on σ. Then, the constrained extensions of σ, denoted $L^c_\sigma(\mathcal{T}')$, is $L^c_{(q,v)}(\mathcal{T}')$. Thus a CTTS can be viewed as working in two "modes": *tracking* (corresponding to the unconstrained behaviour) and *advising* (corresponding to constrained behaviour). Finally, we say $\mathcal{T}'$ is *complete* w.r.t. a timed language $L \subseteq T\Sigma^*$ if $L \subseteq L(\mathcal{T})$.

Definition 4. *A* conflict-tolerant controller *C for $\mathcal{B}$ is a conflict-tolerant TTS over Σ that is complete with respect to $L(\mathcal{B})$.*

The controller C is valid *wrt $\mathcal{B}$ if*

- *C is* non-restricting*: If $\sigma e \in L(\mathcal{B})$ for some environment event $e \in \Sigma_e$, then $\sigma e \in L^c_\sigma(C)$. Thus the controller must not restrict any environment event e enabled in the base system after* any *base system behaviour σ.*

- *C is* non-blocking*: If $\sigma \in L(\mathcal{B})$, then $L^c_\sigma(\mathcal{B}\|C) \neq \{\varepsilon\}$. Thus the controller must not block the base system after* any *base system behaviour σ.*

We now describe when a conflict-tolerant controller satisfies a given conflict-tolerant specification with respect to a given base system $\mathcal{B}$.

Definition 5. *Let f be a conflict-tolerant specification in the form of an advice function over Σ. A conflict-tolerant controller C for $\mathcal{B}$* satisfies *f if for each $\sigma \in L(\mathcal{B})$, $L^c_\sigma(\mathcal{B}\|C) \subseteq f(\sigma)$. Thus after* any *base system behaviour σ, if the base system follows the advice of C, then the resulting behaviour conforms to the safety language $f(\sigma)$.*

Fig. 2(c) and (d) show two conflict-tolerant controllers for the example base system in Fig. 2(a). In the figure we use the convention that the "not-advised" transitions are shown with dotted arrows, and the advised *tcp′* conditions are shown in the state (the tracking *tcp* conditions are all true). The first tolerant controller's behaviour is to advise a **rel** event 1 time unit after the last **rel** event, no matter when it took place. On the other hand, the second tolerant controller in part (d) also advises a **rel** event after 1 time unit if the last two **rel** events were *exactly* 1 time unit apart or "punctual"; but if the last two **rel** events were *less* than 1 time unit apart ("early") it advises a **rel** event after 2 time units, and if the last two **rel** events were *more* than 1 time unit apart ("late") it advises a **rel** event after 0.5 time units. The states of the controller are marked "P", "E", and "L" for "Punctual", "Early", and "Late" respectively, and keep track of this property for the last two **rel** events.

We note that as *classical* controllers, both the controllers have the same effect as the classical controller in part (b): behaviours that are controlled according to their advice will always be (prefixes of) behaviours of the form $(1 \cdot \mathbf{rel})^*$. However as *tolerant* controllers their behaviours are quite different. We will return to these examples in the next section when we illustrate our logic for specifying tolerant specifications.

4 Conflict-Tolerant Metric Interval Temporal Logic

In this section we present our logic for specifying real-time conflict-tolerant specifications, called CT-MITL. Our logic is based on the well-known timed temporal logic Metric Interval Temporal Logic (MITL) [2] which is a popular logic for specifying real-time properties.

We begin by recalling the syntax and semantics of MITL. The syntax of an MITL formula over an alphabet Σ is given by:

$$\theta ::= \top \mid \bot \mid a \mid \neg\theta \mid \theta \vee \theta \mid \theta U_I \theta \mid \theta S_I \theta$$

where $a \in \Sigma$ and I is a non-singular interval with end points in $\mathbb{Q}_{\geq 0} \cup \{\infty\}$.

We interpret formulas of MITL in a "continuous" manner over timed words. Let σ be a timed word, t a timepoint in $[0, dur(\sigma)]$, and θ an MITL formula. Then we define the satisfaction relation $\sigma, t \models \theta$, as follows. We always have $\sigma, t \models \top$ and $\sigma, t \not\models \bot$. In addition we have:

$$\begin{array}{lll}
\sigma, t \models a & \text{iff} & \sigma(t) = a \\
\sigma, t \models \theta_1 U_I \theta_2 & \text{iff} & \exists t'' > t : t'' - t \in I \text{ and } \sigma, t'' \models \theta_2 \text{ and} \\
 & & \forall t' : t < t' < t'', \sigma, t' \models \theta_1 \\
\sigma, t \models \theta_1 S_I \theta_2 & \text{iff} & \exists t'' < t : t - t'' \in I \text{ and } \sigma, t'' \models \theta_2 \text{ and} \\
 & & \forall t' : t'' < t' < t, \sigma, t' \models \theta_1,
\end{array}$$

with the boolean operators interpreted in the standard way. We note that we have used versions of the "Until" and "Since" operators that are "strict" in both the arguments, as is often done when the underlying time domain is dense.

We will make use of the following derived operators: $\delta = \neg \bigvee_{a \in \Sigma} a$, $\phi S \psi = \phi S_{[0,\infty)} \psi$, $init = \neg(\top S \top)$ (*init* thus characterizes the timepoint 0), $\Diamond_I \theta \equiv \top U_I \theta$, $\Box_I \theta \equiv \neg \Diamond_I \neg\theta$, $\diamondsuit_I \theta \equiv \top S_I \theta$, and $\boxminus_I \theta \equiv \neg \diamondsuit_I \neg\theta$.

For an MITL formula θ and a timed word σ, we say $\sigma \models \theta$ iff $\sigma, 0 \models \theta$. We set $L(\theta) = \{\sigma \in T\Sigma^* \mid \sigma \models \theta\}$.

The "past" fragment of MITL is obtained by disallowing the U operator. We will make use of the "strict past" fragment, denoted MITL$_P$, which are past MITL formulas which are not a boolean combination of formulas, one of which is of the form a for some $a \in \Sigma$. Thus an MITL$_P$ formula cannot refer to the current time point. For an MITL$_P$ formula φ, we will say $\sigma \models \varphi$ to mean $\sigma, dur(\sigma) \models \varphi$.

We can now introduce the logic CT-MITL.

Definition 6. *A* Conflict-Tolerant MITL *(*CT-MITL*) formula over an alphabet* Σ *is an* MITL *formula* θ *over* Σ *of the form*

$$\Box \bigwedge_{i \in \{1,\ldots,n\}} (\varphi_i \implies \psi_i)$$

where $n \geq 0$*, and for each* $i \in \{1, \ldots, n\}$*,* φ_i *is an* MITL$_P$ *formula and* ψ_i *is of the form* $\bigvee_{c \in X_i} c$ *for some* $X_i \subseteq \Sigma \cup \{\delta\}$*.*

The CT-MITL formula θ given above induces the immediate advice function $g_\theta : T\Sigma^* \to 2^{\Sigma \cup \{\delta\}}$ given by:

$$g_\theta(\sigma) = \bigcap_{i \in \{1,\ldots,k\},\ \sigma \models \varphi_i} X_i.$$

We use the convention that the intersection of an empty collection of sets is the universal set, in this case $\Sigma \cup \{\delta\}$. Thus, if σ is such that none of the φ_i's is true at the end of σ, then the immediate advice given by g_θ on σ is the set $\Sigma \cup \{\delta\}$. Finally, we will write f_θ to denote the corresponding advice function f_{g_θ}.

We now give a few examples to illustrate the logic. Returning to our example base system of Fig. 2(a), we give a couple of conflict-tolerant specifications in CT-MITL. For convenience we use notation of the form "≥ 1" to denote the interval $[1, \infty)$.

Example 1. *The* CT-MITL *formula* θ_1 *below specifies an immediate advice function "advise a* **rel** *event if one or more time units have elapsed since the last* **rel** *or init event, and time elapse otherwise."*

$$\Box(((\neg \mathbf{rel})\, S_{\geq 1}\, (\mathbf{rel} \vee \mathit{init})) \implies \mathbf{rel}) \wedge ((\diamondsuit_{<1}\, (\mathbf{rel} \vee \mathit{init})) \implies \delta)).$$

The tolerant controller of Fig. 2(c) can be seen to satisfy this conflict-tolerant specification w.r.t. the base system of Fig. 2(a). □

Example 2. *The* CT-MITL *formula* θ_2 *below advises a* **rel** *event after 1 time unit (respectively 2 time units and 0.5 time units) depending on whether the last two* **rel** *events were 1 time unit apart ("Punctual"), less than 1 time unit apart ("Early"), or more than 1 time unit apart ("Late").*

$$\begin{aligned}\Box(\quad & (((\neg \mathbf{rel}) S_{\geq 1} P) \implies \mathbf{rel}) \wedge (((\neg \mathbf{rel}) S_{<1} P) \implies \delta) \wedge \\ & (((\neg \mathbf{rel}) S_{\geq 2} E) \implies \mathbf{rel}) \wedge (((\neg \mathbf{rel}) S_{<2} E) \implies \delta) \wedge \\ & (((\neg \mathbf{rel}) S_{\geq 0.5} L) \implies \mathbf{rel}) \wedge (((\neg \mathbf{rel}) S_{<0.5} L) \implies \delta)),\end{aligned}$$

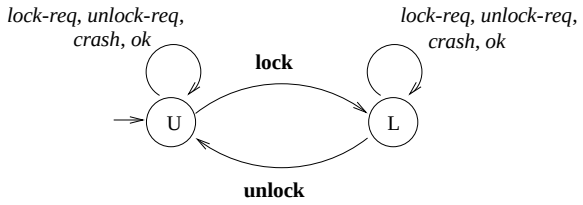


Figure 3: The door motor base system.

where

$$\begin{aligned} P &= \mathit{init} \vee (\mathbf{rel} \wedge ((\neg \mathbf{rel}) S_{=1} \mathbf{rel})) \\ L &= \mathbf{rel} \wedge ((\neg \mathbf{rel}) S_{>1} \mathbf{rel})) \\ E &= \mathbf{rel} \wedge (\diamondsuit_{<1} \mathbf{rel})). \end{aligned}$$

The tolerant controller of Fig. 2(d) can be seen to satisfy the tolerant specification given by this formula. □

It is interesting to note that viewed as classical MITL specifications, both θ_1 and θ_2 describe *exactly* the same timed language: $L(\theta_1) = L(\theta_2) = (1 \cdot \mathbf{rel})^*_{\leq}$. However, as illustrated by the examples, the tolerant specifications induced by θ_1 and θ_2 are very different. The controllers in Fig. 2 each satisfies its respective specification, but not the other's. Further, for an CT-MITL formula θ, we always have $L(\theta) = f_\theta(\varepsilon)$. Thus the timed language denoted by θ corresponds to the initial safety cone of the tolerant spec denoted by θ.

As a final example we look at a couple of features for a car door motor system, adapted from [4]. The base system here (shown in Fig. 3) models a car door motor, which has environment events $\Sigma_e = \{\mathit{lock\text{-}req}, \mathit{unlock\text{-}req}, \mathit{crash}, \mathit{ok}\}$ and system events $\Sigma_s = \{\mathbf{lock}, \mathbf{unlock}\}$. The "Overheat Protection" feature aims to protect the motor from overheating due to the excessive locking and unlocking of doors within a short period of time. It recommends that if two **lock** events occur within 30 sec the system events **lock** and **unlock** be disabled for 180 sec.

Example 3. *A possible tolerant specification for this feature is:*

$$\Box((\diamondsuit_{\leq 180}(\mathbf{lock} \wedge \diamondsuit_{\leq 30}\mathbf{lock})) \implies \delta \vee \mathit{env}).$$

Here env stands for the formula $\bigvee_{e \in \Sigma_e} e$. *Fig: 4 shows a possible tolerant controller that satisfies this specification.* □

Example 4. *The "Crash Safety" feature for the door motor requires that if a crash occurs, then unless an ok event occurs the door must be unlocked within* 15 *seconds, and kept unlocked till an ok event occurs. A possible tolerant specification for this feature is:*

$$\begin{aligned} \Box(\quad & ((\neg(\mathit{ok} \vee \mathbf{unlock})\, S_{<15}\, \mathit{crash}) \implies (\delta \vee \mathbf{unlock} \vee \mathit{env})) \wedge \\ & ((\neg(\mathit{ok} \vee \mathbf{unlock})\, S_{\geq 15} \mathit{crash}) \implies (\mathbf{unlock} \vee \mathit{env}))). \end{aligned}$$

Fig: 5 shows a tolerant controller satisfying the crash safety specification above. □

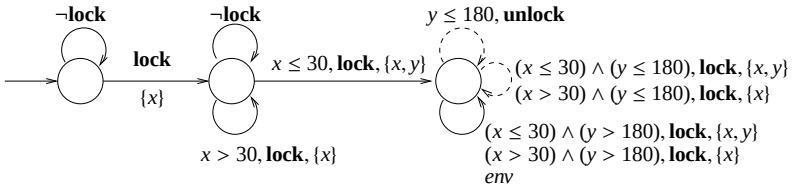


Figure 4: A controller for overheat protection.

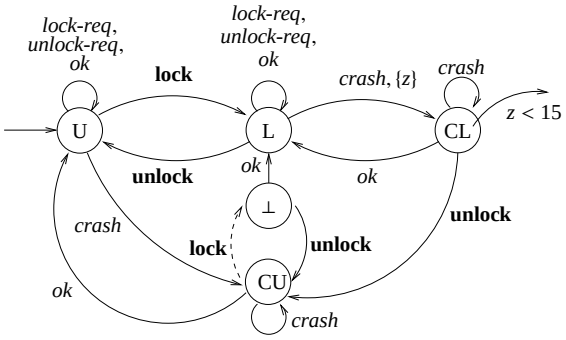


Figure 5: A controller for crash safety protection.

We can now define the natural verification and synthesis problems for the logic CT-MITL.

Definition 7. *(*Verification Problem for CT-MITL*) Given a base system $\mathcal{B}$ over Σ, a conflict-tolerant controller C for $\mathcal{B}$, and a* CT-MITL *formula θ, check whether C is a valid conflict-tolerant controller for $\mathcal{B}$ which satisfies the advice function f_θ.*

Definition 8. *(*Synthesis Problem for CT-MITL*) Given a base system $\mathcal{B}$ over Σ, and a* CT-MITL *formula θ, check whether there exists a valid conflict-tolerant controller for $\mathcal{B}$ which satisfies the advice function f_θ. If so, construct one.*

5 Monitoring Timed Transition System for MITL_P

The main step in solving the verification and synthesis problems for CT-MITL is the construction of a "monitoring" TTS for a given CT-MITL specification. Without loss of generality we assume that the specification uses only the modalities untimed S and $\diamondsuit$. The modality S_I can be expressed in terms of $\diamondsuit$ and $\boxminus$ as follows:

$$\varphi_1 S_I \varphi_2 \equiv \begin{cases} \diamondsuit_I \varphi_2 \wedge \boxminus_{[0,l)}(\varphi_1 \wedge (\varphi_1 S \varphi_2)) & \text{if } I = [l, r\rangle, l > 0 \\ \diamondsuit_I \varphi_2 \wedge \boxminus_{[0,l]}(\varphi_1 \wedge (\varphi_1 S \varphi_2)) & \text{if } I = (l, r\rangle, l > 0 \\ \diamondsuit_I \varphi_2 \wedge \varphi_1 S \varphi_2 & \text{if } I = \langle 0, r\rangle \end{cases}$$

We also assume the standard notion of subformulas: thus the subformulas of the formula $\varphi = aS(\diamondsuit_{[1,2]}(b \vee \diamondsuit_{(0,\infty)}c))$ are φ, a, $\diamondsuit_{[1,2]}(b \vee \diamondsuit_{(0,\infty)}c)$, $b \vee \diamondsuit_{(0,\infty)}c$, b, $\diamondsuit_{(0,\infty)}c$ and c. The *positive subformulas* of a formula φ are all its subformulas which are not of the form $\neg\psi$ and we denote the set of all positive subformulas of φ by $psf(\varphi)$.

A *monitoring guard* g over a set of states Q and clocks C is a boolean combination of configurations constraints and constraints of the form "a" for $a \in \Sigma$. A monitoring guard g is evaluated over a triple (c, q, v), where $c \in \Sigma \cup \{\delta\}$, $q \in Q$, and v is a valuation over C, as follows: $(c, q, v) \models g$ iff g is a configuration constraint and $(q, v) \models g$; and $(c, q, v) \models a$ iff $c = a$.

Let φ be a MITL$_P$ formula. A *monitoring* TTS (MTTS) for φ is a pair $(\mathcal{T}_\varphi, g_\varphi)$ where $\mathcal{T}_\varphi$ is a TTS with configuration constraints and g_φ is monitoring guard wrt $\mathcal{T}_\varphi$, such that $\forall\sigma \in T\Sigma^*, \sigma \models \varphi \iff (last(\sigma), config_{\mathcal{T}}(\sigma)) \models g$.

Lemma 1. *Given an* MITL$_P$ *formula* φ *we can effectively construct an MTTS* $(\mathcal{T}_\varphi, g_\varphi)$ *for* φ.

Proof. We prove this lemma by induction on the structure of φ. The only interesting cases are $\psi_1 S \psi_2$ and $\diamondsuit_I \psi$. Let $\varphi = \psi_1 S \psi_2$. By induction hypothesis there exists an MTTS $\mathcal{T}_{\psi_1}$ and $\mathcal{T}_{\psi_2}$ with g_{ψ_1} and g_{ψ_2} are the monitoring guards. Let $\mathcal{T}_{\psi_1 S \psi_2}$ be the OpenTTS as shown in Fig: 6. Then the TTS $\mathcal{T}_{\psi_1 S \psi_2} \parallel \mathcal{T}_{\psi_1} \| \mathcal{T}_{\psi_2}$ is a monitoring TTS for $\psi_1 S \psi_2$ with $g_{\psi_1 S \psi_2} = (q_1 \wedge (x > 0)) \vee (q_2 \wedge (x = 0))$.

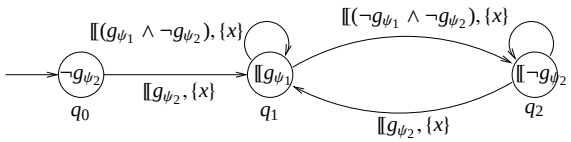


Figure 6: OpenTTS $\mathcal{T}_{\psi_1 S \psi_2}$.

As an exercise let us construct an MTTS $\mathcal{T}_\varphi$ for the formula $\varphi = \diamondsuit_{(l,r)}a$ before dealing with the general formula $\diamondsuit_I \psi$. Our construction for φ is based on the inductive construction given in [2] and [7]. For easy understanding we begin with a brief description of $\mathcal{T}_\varphi$. $\mathcal{T}_\varphi$ reset clocks x and y when a occurs for the first time. When a becomes true next time if $y < r - l$ then we reset y again as we can safely ignore the fact that the formula a has been false since the previous reset of y and if $y \geq r - l$ we reset a fresh pair of clocks (see [2] for details and Fig: 7 for an illustration). Once the value of the clock y goes beyond r, the values of the clocks x and y are of no use and can therefore be reused. Thus $\mathcal{T}_\varphi$ needs to reset at most a pair of clocks in any interval of size $r - l$ and therefore we conclude that the maximum no. of clocks required by $\mathcal{T}_\varphi$ is at most $2 * \lceil r/(r-l) \rceil$. And the monitoring guard of $\mathcal{T}_\varphi$ is given by $g_\varphi = \bigvee_{i=1}^{i=n} ((x_i > l) \wedge (y_i < r))$ where $n = 2 * \lceil r/(r-l) \rceil$. As an example MTTS $\mathcal{T}_{\diamondsuit_{(1,2)}a}$ along with the monitoring guard $g_\varphi = \bigvee_{i=1}^{i=2} ((x_i > 1) \wedge (y_i < 2))$ shown in the Fig: 8.

Let us now construct a monitoring OpenTTS for the formula $\diamondsuit_{(l,r)}\psi$ [1]. As a first step, for a monitor guard g let us introduce the notation for the *left-closure* of g, denoted

[1] MTTS for other cases can be constructed similarly.

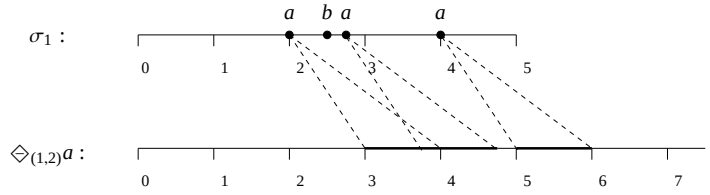


Figure 7: Truth value for $\diamondsuit_{(1,2)}a$ over σ.

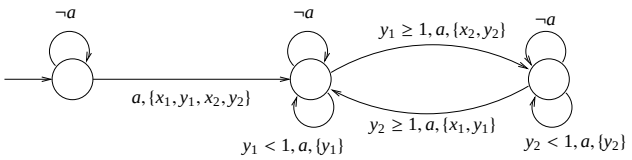


Figure 8: $\mathcal{T}_{\diamondsuit_{(1,2)}a}$.

$[\![g$.

$$
\begin{aligned}
[\![u &= u, u \in Q \cup \Sigma,\\
[\![(x \rtimes c) &= x \geq c, \rtimes \in \{>, \geq\},\\
[\![(x \ltimes c) &= x \ltimes c, \ltimes \in \{<, \leq\},\\
[\![(g_1 \vee g_2) &= [\![g_1 \vee [\![g_2,\\
[\![(g_1 \wedge g_2) &= [\![g_1 \wedge [\![g_2,
\end{aligned}
$$

where c is a rational constant. Symmetrically we can define the notation $g]\!]$. For a monitoring g, we define $[\![g]\!] = ([\![g) \vee (g]\!])$.

Coming back to the construction, by induction hypothesis there exists an MTTS $\mathcal{T}_\psi$ for ψ. Let $n = \lceil r/(r-l) \rceil$ and for each $i \in \{1, \ldots, n\}$ let $f_i = \neg g_\psi, \{y_i\}$ and $g_i = ([\![g_\psi) \wedge (y_i < r - l), \{y_i\}$, and for each $i \in \{1, \ldots, n-1\}$ let $h_i = ([\![g_\psi) \wedge (y \geq r - l), \{x_{i+1}, y_{i+1}\}$. Let $\mathcal{U}_\varphi$ be the OpenTTS shown in Fig: 9. Then $(\mathcal{U}_\varphi \| \mathcal{T}_{\psi_1} \| \mathcal{T}_{\psi_2}, g_\varphi)$ is an MTTS for φ with $g_\varphi = \bigvee_{i=1}^{i=n}((x_i > l) \wedge (y_i < r))$.

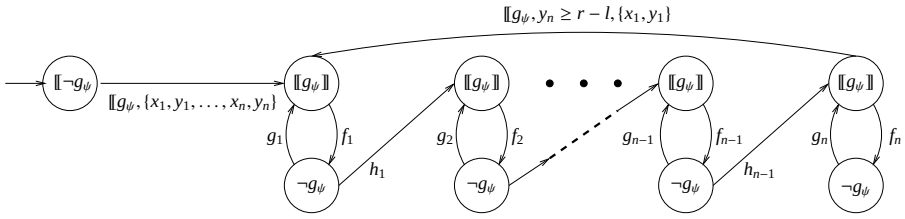


Figure 9: OpenTTS for the formula $\diamondsuit_{(l,r)}\psi$.

5.1 Our Construction

In this section we propose an alternate construction for $\mathcal{T}_\varphi$ when $\varphi = \diamondsuit_{\langle l,r\rangle}\psi$. We call our construction *delay-then-extend* or DTE for short. Our construction uses an optimal number of clocks and in general it uses one clock less than the constructions in [2, 7].

So let us begin with a brief discussion of $\mathcal{T}_\varphi$ which monitors the formula $\diamondsuit_{(l,r)}a$. $\mathcal{T}_\varphi$ reset clocks x, called a *leading clock*, and y, called the *trailing clock of* x, when a occurs for the first time. When a becomes true next time if clock value $y < r - l$ then $\mathcal{T}_\varphi$ resets y again. Thus when a clock $x'(\neq y)$ is reset next time the clock value of y is at least $r - l$ and as we shall see, this fact is crucially used in our construction. When the clock value of y becomes l, $\mathcal{T}_\varphi$ resets a clock z, which remains active until it turns $r - l$. This enables the TTS to reuse the clocks x and y as soon as value of y becomes l and the clock z when its value becomes $r - l$. As the trailing clock of x' becomes l the clock value of z will be at least $r - l$ because when x' was reset the value of y was at least $r - l$ so the clock z can be reused. Fig: 10 illustrates our idea for the formula $\varphi = \diamondsuit_{(1,2)}a$. $\mathcal{T}_{\diamondsuit_{(1,2)}a}$ with $g_\varphi = \bigvee_{i=1}^{i=2}((x_i > 1) \wedge (\bigvee_{i=1}^{i=4}(p_i \wedge (z < 1))))$ and $\Sigma' = \Sigma - \{a\}$ is shown in Fig: 11.

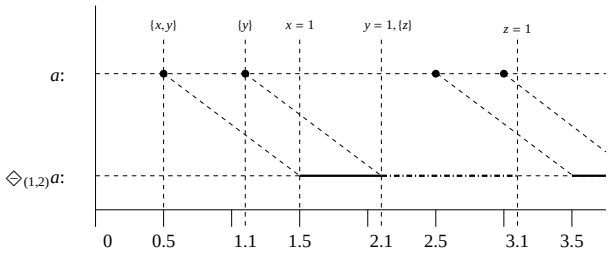


Figure 10: An illustration of our idea.

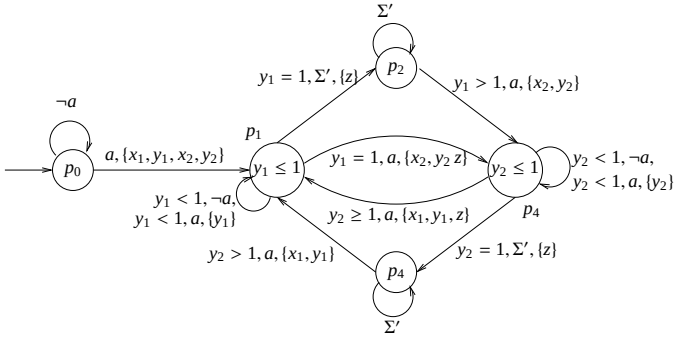


Figure 11: Monitoring TTS $\mathcal{T}_{\diamondsuit_{(1,2)}a}$.

Thus, as $\mathcal{T}_\varphi$ needs to reset at most a pair of clocks in an interval of size $r - l$, we

conclude that the maximum no. of clocks required by $\mathcal{T}_\varphi$ is at most $2 * \lceil l/(r-l) \rceil + 1$. It is easy to extend the above construction to any given MITL$_P$ formula which is of the form $\diamondsuit_{\langle l,r \rangle}\psi$. Thus we conclude that:

Theorem 1. *For any* MITL$_P$ *formula* $\diamondsuit_{\langle l,r \rangle}\psi$ *there exists a monitoring* TTS *for it which uses at most* $2 * \lceil l/(r-l) \rceil + 1$ *clocks.*

Let us now compare the monitoring TTS $\mathcal{T}_A$ for formula φ given by AFH construction with the monitoring TTS $\mathcal{T}_D$ given by our construction. In $\mathcal{T}_A$, φ is true at any point if the guard $(x > l) \wedge (y < r)$ is satisfied at that point. In our construction as we reset a clock z when the clock value of y becomes l the formula φ is true at any point if the guard $((x > l) \wedge (y < l)) \vee (q \wedge (z < r - l))$, where q any state in T_D except possibly the initial state, evaluates to true at that point. In $\mathcal{T}_D$, we use $2 * \lceil l/(r-l) \rceil + 1$ $(= 2n + 1)$ clocks while $\mathcal{T}_A$ uses $2n + 2$ clocks. Thus the region automaton of $\mathcal{T}_D$ has lesser no. of states than the region automaton of $\mathcal{T}_A$. Thus from the verification point of view DTE is more efficient than AFH. Let us now superficially compute the size of the region automaton yield by $\mathcal{T}_D$ and $\mathcal{T}_A$. In $\mathcal{T}_D$ all but one clocks are compared with only l and one clock is compared with $r - l$. Thus the no. of regions induced by the clocks in $\mathcal{T}_D$ is at most $c_1 = 2nl \cdot 2^n \cdot (2l+2)^{2n}$. In $\mathcal{T}_A$ $(n+1)$ clocks are compared with l and the rest of the clocks, i.e. $(n+1)$ clocks, are compared with r and therefore the no. of regions induced by the clocks in $\mathcal{T}_A$ is at most $c_2 = (2n+2) \cdot 2^{n+1} \cdot (2l+2)^n \cdot (2r+2)^n$. Thus the space complexity of $\mathcal{T}_D$ is $O(c_1)$ while that of $\mathcal{T}_A$ is $O(c_2)$.

5.2 Optimality of DTE construction

In this section we prove that our construction is optimal in terms of no. of clocks used, i.e. there exists a formula of the form $\diamondsuit_{\langle l,r \rangle}\psi$ such that any monitoring TTS for it will have at least $2 * \lceil l/(r-l) \rceil + 1$ clocks. We do this by showing that there exists timed word σ such that in order to monitor σ for the formula $\diamondsuit_{(l,r)}a$ for some $a \in \Sigma$ any monitoring TTS requires at least $2 * \lceil l/(r-l) \rceil + 1$ clocks.

Let us fix some $a \in \Sigma$ and let $\varphi = \diamondsuit_{(l,r)}a$. For the purpose of contradiction lets us assume that there exists an MTTS $(\mathcal{T}_\varphi, g_\varphi)$ for φ which uses $2 * \lceil l/(r-l) \rceil$ $(= 2n)$ clocks viz, $x_1, y_1, \cdots, x_n, y_n$. Let C be the set $\{x_1, y_1, \cdots, x_n, y_n\}$ and let Q be the set of states of $\mathcal{T}_\varphi$. Without loss of generality let us assume that $l, r \in \mathbb{N}$ and $\mathcal{T}_\varphi$ uses only integer constants. Now let σ be a timed over $\{a\}$ satisfying the following conditions:

- $d_0 = 0$, $|\sigma|$(no.of action points in σ)$=2n+2$ and $dur(\sigma) = \Sigma_{i=0}^{i=2n+1} d_i$.
- for each $i \in \{0, \ldots, n\}$, $t_{2i+1} - t_{2i} = \epsilon$ and for each $j \in \{1, \ldots, n\}$, $t_{2i} - t_{2i-1} = r - l + \epsilon$.

where $t_i = \Sigma_{i=0}^{i=2n+1} d_i$ and $0 < \epsilon < (r-l)/(2n+2)$.

Let ρ to be the unique run of $\mathcal{T}_\varphi$ on σ an let $v_t(x)$ be the valuation of clock x at time t along this run.

Definition 9. *For any* $t, u \in [0, dur(\sigma)]$, $IDiff_t(u) = \{x \in C \mid (u - t) - v_u(x) \in \mathbb{Z} - \{0\}\}$.

Lemma 2. *For any two time points* $t_1, t_2 \in [0, \ldots, dur(\sigma)]$ *if for all* $t \in (t_1, t_2)$ $IDiff_t(t) = \emptyset$ *then for all monitoring guards* g *over* C *and* Q, $\sigma, t \models g$ *for some* $t \in (t_1, t_2)$ *iff* $\sigma, t \models g$ *for all* $t \in (t_1, t_2)$.

As $\mathcal{T}_\varphi$ is deterministic lemma 2 implies that for every $i \in \{0, \cdots, 2n-1\}$, $IDiff_{t_i}(t_i + l) \neq \emptyset$ and for every point t if $\mathcal{T}_\varphi$ does an ϵ-transition at t on σ then $IDiff_t(t) \neq \emptyset$. Then from the above facts and from the definition of *IDiff* one can deduce that:

Lemma 3. *For any two distinct points $t_1, t_2 \in \{t_0, \ldots, t_{2n+1}\}$, $IDiff_{t_1}(u) \cap IDiff_{t_2}(u) = \emptyset$ for all $u \in \mathbb{R}_{\geq 0}$.*

Lemma 4. *Let $u = t_i + l$ for some $0 \leq i \leq 2n+1$ and let $x \in IDiff_{t_i}(u)$. Then $x \in IDiff_{t_i}(t_i + t)$ for all $t \in (u - v_u(x), u]$.*

Proof. Let $w = u - v_u(x)$ be the last point before u where the clock x is reset. As $x \in IDiff_{t_i}(u)$, $w - t_i \in \mathbb{N}^*$ and for any $t \in (u - v_u(x), u]$, $t(x) = t - w$. This implies $x \in IDiff_{t_i}(t)$ as $(t - t_i) - (t - w) = w - t_i \in \mathbb{Z} - \{0\}$. □

Corollary 1. *$IDiff_{t_i}(t_i + t) \neq \emptyset$ for all $t \in (t_i, u]$.*

Proof. As the clock x is reset at w, due to the properties of σ, w is point of ϵ-transition in the run ρ of $\mathcal{T}_\varphi$ on σ. Then as mentioned earlier there exists a clock such that $y \in IDiff_{t_i}(w)$ and therefore by lemma 4 for all $t \in (w - v_w(y), w]$, $y \in IDiff_{t_i}(t)$. Note that $v_w(y) > 0$. Let w' be the last point before w where the clock y is reset. Again due to σ, w' is point of ϵ-transition in the run ρ. Thus we have that $IDiff_{t_i}(t_i + t) \neq \emptyset$ for all $t \in (t_i, u]$. □

Let t be a point such that $t_{2n+1} < t < n * (r - l)$. Then by lemma 3 and corollary 1 we have that for every $i \in \{1, \cdots, 2n+1\}$, $IDiff_{t_i}(t) \neq \emptyset$. As $\mathcal{T}_\varphi$ uses at most $2n$ clocks it follows that there exist distinct $i, j \in \{1, \ldots, t_{2n+1}\}$ that $IDiff_{t_i}(t) \cap IDiff_{t_j}(t) \neq \emptyset$, which is a contradiction by lemma 3. Thus we have:

Theorem 2. *DTE is optimal in terms of no. clocks.*

5.3 An Example MTTS Construction

Example 5. *Let us construct an MTTS for the formula $\varphi = \diamondsuit_{\leq 180}(\textbf{\textit{lock}} \wedge \diamondsuit_{\leq 30}\textbf{\textit{lock}})$. Recall that φ is the premise of the specification for the feature overheat protection given in section 4. As a first step let us first compute the set $psf(\varphi)$.*

$$psf(\varphi) = \{\diamondsuit_{\leq 180}(\textbf{\textit{lock}} \wedge \diamondsuit_{\leq 30}\textbf{\textit{lock}}), \textbf{\textit{lock}} \wedge \diamondsuit_{\leq 30}\textbf{\textit{lock}}, \diamondsuit_{\leq 30}\textbf{\textit{lock}}, \textbf{\textit{lock}}\}.$$

Now it is not difficult to see that the TTS *shown in Fig: 12 is an MTTS for $\diamondsuit_{\leq 30}\textbf{\textit{lock}}$ with $g_{\diamondsuit_{\leq 30}\textbf{\textit{lock}}} = \mathcal{T}_{\diamondsuit_{\leq 30}\textbf{\textit{lock}}}.(p_1 \wedge (x \leq 30))$. Let $g = g_{\diamondsuit_{\leq 30}\textbf{\textit{lock}}}$ and let $\mathcal{U}_\varphi$ be the* OpenTTS *shown in Fig: 13. Then $\mathcal{U}_\varphi \| \mathcal{T}_{\diamondsuit_{\leq 30}\textbf{\textit{lock}}}, g_\varphi$ is an MTTS for φ with $g_\varphi = \mathcal{U}_\varphi.(p_1 \wedge (x \leq 180))$.* □

6 Verification

We can now solve the verification problem for CT-MITL using the construction of a monitoring TTS in the previous sections.

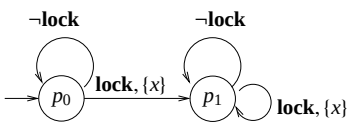


Figure 12: $\mathcal{T}_{\Diamond_{\leq 30}\textbf{lock}}$.

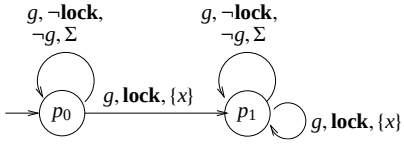


Figure 13: OpenTTS $\mathcal{U}_\varphi$.

Theorem 3. *The verification problem for* CT-MITL *can be solved in EXPSPACE.*

Proof. Let $\mathcal{B}$ a base system over an alphabet Σ, C a conflict tolerant controller and θ a CT-MITL specification. It is easy to see that a necessary and sufficient condition for C to be a valid controller for $\mathcal{B}$ and satisfying θ, is to check that in the synchronized product $\mathcal{B}\|C\|\mathcal{T}_\theta$, there does *not* exist a configuration $((b, p, q), v)$ which is reachable from the initial configuration $((s_\mathcal{B}, s_C, s_{\mathcal{T}_\theta}), \vec{0})$, and satisfies one of the following conditions:

1. C is restricting: there exists $e \in \Sigma_e$ such that a e transition is enabled at (b, v) in $\mathcal{B}$ but is not allowed by C at (p, v)
2. C is blocking: there is no discrete or delay transition that is enabled at $((b, p), v)$. Recall our assumption that $\mathcal{B}$ is non-blocking.
3. C does not satisfy θ: There exists $c \in \Sigma \cup \{\delta\}$ which is enabled at $((b, p), v)$ in $\mathcal{B} \parallel C$ but $c \notin \bigcap_{i\in\{1,\ldots,n\},\ (q,v)\models g_{\varphi_i}} X_i$.

All the conditions above can be checked in EXPSPACE using the *region graph* [1] of the synchronized product $\mathcal{B}\|C\|\mathcal{T}_\theta$. □

7 Synthesis

In this section we synthesise a conflict-tolerant controller and prove that the controller meets the specification given by the user. So let first define what we mean by the composition of conflict-tolerant controller. Let $C_1 = (\mathcal{T}_1, N_1, tcp_1)$ and $C_2 = (\mathcal{T}_2, N_2, tcp_2)$ be two conflict-tolerant controllers. Then the synchronized product $C_1 \parallel C_2$ is given by $(\mathcal{T}_1 \parallel \mathcal{T}_1, N_1 \times N_2, tcp_1 \wedge tcp_2)$.

Let us now give a brief account of how we synthesise a conflict-tolerant controller for a given CT-MITL specification $\theta = \Box(\bigwedge_{i=1}^{i=n}(\varphi_i \implies \psi_i))$. First we construct a monitoring TTS $\mathcal{T}_{\varphi_i}$ for every formula φ_i occurring in the specification θ. Let $\mathcal{U}_{\varphi_i}$ be the conflict-tolerant OpenTTS shown in Fig: 14 with *tcp* condition $h = \top$ if $\delta \in X_i$ and $\neg g_{\varphi_i}$ otherwise. Let $(C_{\varphi_i}, g_{\varphi_i})$ be the conflict-tolerant TTS given by the composition $\mathcal{U}_{\varphi_i} \parallel \mathcal{T}_{\varphi_i}$. It is not difficult to argue that the resultant controller C_{φ_i} is indeed a conflict-tolerant controller which advises ψ_i whenever φ_i is true and Σ otherwise. Finally we argue that the controller C_θ given by the composition of the controllers $C_{\varphi_1}, \ldots, C_{\varphi_n}$ is a conflict-tolerant controller satisfying θ.

Theorem 4. *Given a base system $\mathcal{B}$ and a* CT-MITL *specification θ over Σ, we can check if it is* feasible *to construct a valid conflict-tolerant controller for $\mathcal{B}$ that satisfies θ.*

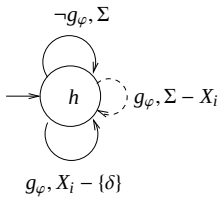


Figure 14: OpenTTS $\mathcal{U}_{\varphi_i}$.

Proof. We claim that there exists a valid controller for $\mathcal{B}$ satisfying θ iff in the synchronized product $\mathcal{B}||C_\theta$, there does not exist a configuration $((b, q), v)$ which is reachable from the initial configuration $((s_\mathcal{B}, s_{C_\theta}), \vec{0})$, and satisfies one of the following conditions:

1. C_θ is restricting: there exists $e \in \Sigma_e$ such that a e transition is enabled at (b, v) in $\mathcal{B}$ but is not advised by C_θ at (p, v)

2. C is blocking: there is no discrete or delay transition that is enabled at $((b, p), v)$. Recall our assumption that $\mathcal{B}$ is non-blocking.

If such a configuration $((b, q), v)$ exists, then clearly a controller cannot be valid for $\mathcal{B}$ and satisfy C_θ at the same time. Conversely, if no such $((b, q), v)$ exists, then C_θ itself is a valid controller for $\mathcal{B}$ that satisfies $\mathcal{T}_\theta$.

The condition above can be checked using the region graph [1] $\mathcal{R}$ for $\mathcal{B}||\mathcal{T}_\theta$. We can check whether a bad configuration satisfying one of the conditions above is reachable in $\mathcal{R}$ in time linear in the size of $\mathcal{R}$. The size of the region graph is exponential in the number of clocks and the maximal constants in the guards of $\mathcal{B}||\mathcal{T}_\theta$.

□

Let us now return to the example given in section 5.3 to illustrate our construction. Consider the conflict-tolerant OpenTTS $\mathcal{E}_1$ shown in Fig: 15.

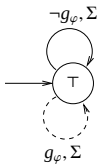


Figure 15: OpenTTS $\mathcal{E}_1$.

Since the guard g_φ is the monitoring guard for φ in the OpenTTS $\mathcal{U}_{\Diamond_{\leq 30}\mathbf{lock}}$ it is straightforward to see that the CTTS given by $\mathcal{E}_1||\mathcal{U}_\varphi \parallel \mathcal{U}_{\Diamond_{\leq 30}\mathbf{lock}}$ meets the specification $\varphi \implies \delta$.

8 Signal Controller Synthesis

In this section we discuss the synthesis of conflict-tolerant controllers using the widely used signal porcessing circuits. The advantage of this method over the automata-theoretic approach is that the controller implementation is more user friendly as it can be made with off-the-shelf electrical circuits. As a result the conflict-tolerant controller synthesis is much more simpler and efficient. We will not discuss the implementation details of the circuits as it outside the scope of this work.

A binary signal s can be viewed as a map from a closed interval $[0, r]$ to the set $\{0, 1\}$. We call r the *duration of* s and we denote it by $dur(s)$. We define the boolean combinations of two signals with same durations in the expected way. Let $l, k \in \mathbb{Q}_{\geq 0}$. Then we define the *l-delay signal* of s to be the signal u_l^s given by $\forall t \in [0, l),\ u_l^s(t) = 0, \forall t \geq l,\ u_l^s(t) = s(t - l)$ and $dur(u_l^s) = dur(s)$ and the *k-extension signal* of s to be the signal v_k^s given by $\forall t : 0 \leq t \leq dur(s),\ v_k^s(t) = 1 \iff \exists t' \leq k,\ s(t - t') = 1$ and $dur(v_k^s) = dur(s)$. Let ϕ be an MITL$_P$ formula and σ, a timed word over an alphabet Σ. Then by s_ϕ^σ we denote the binary signal s such that $dur(s) = dur(\sigma)$ and $\forall t \in [0, \ldots, dur(\sigma)], \sigma, t \models \phi \iff s(t) = 1$.

A *k-arity signal circuit* is a circuit which takes k binary signals with equal durations as inputs and produces a binary signal as the output whose duration is same as that of its input signals. Let Φ be a k-arity signal circuit and let $s_1, \ldots, s_k$ be the k input signals of Φ with durations r. Then we denote the output signal of Φ at time t by $\Phi(s_1, \ldots, s_k)(t)$ and by $\Phi(s_1, \ldots, s_k)$ we mean the signal s such that $dur(s) = r$ and $\forall t \in [0, \ldots, r], s(t) = \Phi(s_1, \ldots, s_k)(t)$. Let $act = \bigvee_{i=1}^{i=n} a_i$. Then a *monitoring circuit* of an MITL$_P$ formula ϕ is a 1-arity signal circuit Π_ϕ such that for all $\sigma \in T\Sigma^*, \Pi_\phi(s_{act}^\sigma) = s_\phi^\sigma$. And an *open montioring circuit* for ϕ, denoted Φ_ϕ, is inductively defined as follows:

$$
\begin{aligned}
\Phi_a &: \forall \sigma \in T\Sigma^*, \Phi_a(\sigma) = s_a^\sigma, \text{ where } a \in \Sigma, \\
\Phi_{\neg\psi} &: \forall \sigma \in T\Sigma^*, \Phi_{\neg\psi}(s_\psi^\sigma) = s_{\neg\psi}^\sigma, \\
\Phi_{\psi_1 \vee \psi_2} &: \forall \sigma \in T\Sigma^*, \Phi_{\psi_1 \vee \psi_2}(s_{\psi_1}^\sigma, s_{\psi_2}^\sigma) = s_{\psi_1}^\sigma \vee s_{\psi_2}^\sigma, \\
\Phi_{\psi_1 S \psi_2} &: \forall \sigma \in T\Sigma^*, \Phi_{\psi_1 S \psi_2}(s_{\psi_1}^\sigma, s_{\psi_2}^\sigma) = s_{\psi_1 S \psi_2}^\sigma, \\
\Phi_{\diamondsuit_I \psi} &: \forall \sigma \in T\Sigma^*, \Phi_{\diamondsuit_I \psi}(s_\psi^\sigma) = s_{\diamondsuit_I \psi}^\sigma.
\end{aligned}
$$

Lemma 5. *For any* MITL$_P$ *formula* ϕ *we can construct a monitoring circuit* Φ_ϕ *for it.*

Proof. We prove the lemma by structural induction on ϕ. The circuits for the base cases and the boolean combinations are trivial. The only interesting cases are $\psi_1 S \psi_2$ and $\diamondsuit_I \psi$. So lets start with $\phi = \psi_1 S \psi_2$. By induction hypothesis there exist monitoring circuits Π_{ψ_1} and Π_{ψ_2} which output signals $s_{\psi_1}^\sigma$ and $s_{\psi_2}^\sigma$ respectively. Then the figure shown in Fig: 16 is an open monitoring circuit for ϕ which when fed with signals $s_{\psi_1}^\sigma$ and $s_{\psi_2}^\sigma$ outputs the signal s_φ^σ. Now one can easily construct a combinatorial monitoring circuit Π_ϕ for ϕ using $\Pi_{\psi_1}, \Pi_{\psi_2}$ and Φ_ϕ. We skip the implementation details of the circuit.

Let ψ be an MITL$_P$ formula and let $\phi = \diamondsuit_{\langle l,r \rangle} \psi$. Inductively, we assume that the circuit Π_ψ is already constructed and its output signal s_ψ^σ is available to us. Then we construct an open monitoring circuit Φ_ϕ for ϕ using two important elements: l-delay element and $(r - l)$-extension element. Given an input signal s the l-delay element

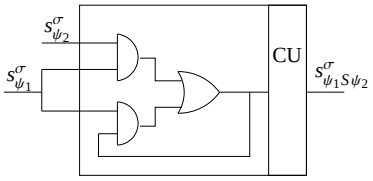


Figure 16: An open monitoring circuit Φ_ϕ for $\psi_1 S \psi_2$.

outputs the signal u^s_l and the k-extension element outputs the signal v^s_k. The schematic diagrams for the l-delay element and $(r-l)$ extension element are given in Fig: 17 and Fig: 18 respectively.

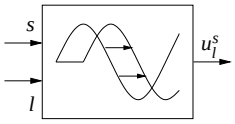


Figure 17: l-delay element.

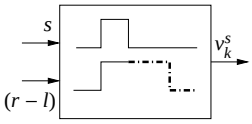


Figure 18: $(r-l)$-extension element.

Then the circuit shown in Fig: 19 is an open monitoring circuit for ϕ which if given the input signal s^{σ}_{ψ} generates s^{σ}_{ϕ} as the output signal. It is not difficult to see that the circuit Φ_ϕ is essentially derived from the DTE constuction we mentioned in section 5.1. Once again we can construct a combinatorial monitoring circuit Π_ϕ for ϕ using Π_ψ and Φ_ϕ.

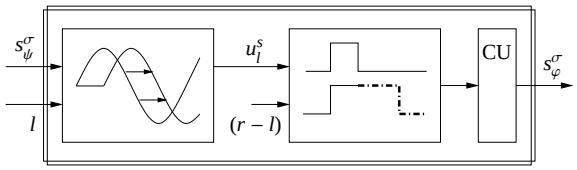


Figure 19: An open monitoring circuit Φ_ϕ for ϕ.

As a simple exercise, a sample input signal to the open monitoring circuit $\Phi_{\diamondsuit_{(1,2)}\psi}$ and its output is shown in Fig: 20. The intermediate signal u^s_ψ is also shown in the figure.

□

A *n-signal demultiplexer* or n-demux is a demultiplexer circuit with one input line and n output lines. For each $i \in \{1,\ldots,n\}$ we denote each line of the n-demux by l_i. Let $\Sigma = \{a_1,\ldots,a_n\}$ and let $\theta = \Box(\bigwedge_{i=1}^{i=n}(\phi_i \implies \psi_i))$ be a CT-MITL specification over Σ. Then a *conflict-tolerant signal demultiplexer* or CT-demux for short, for θ is an $(n+1)$-signal demux such that when given the input signal s^{σ}_{act} activates the output line

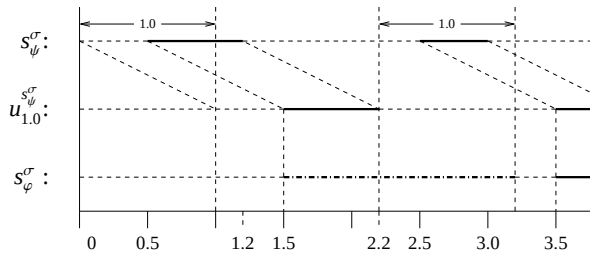


Figure 20: A sample input signal to $\Phi_{\diamondsuit_{(1,2)}\psi}$ and its output signal.

l_i iff $a_i \in \bigcap_{\sigma,t\models\phi_i} X_i$ and l_{n+1} iff $\delta \in \bigcap_{\sigma,t\models\phi_i} X_i$ for all $t \in [0,\ldots,dur(s^{\sigma}_{act})]$. A schematic diagram for a conflict-tolerant signal controller for θ is given in Fig: 21.

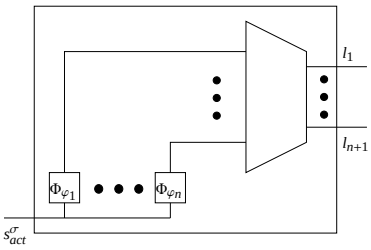


Figure 21: A schematic diagram for a conflict-tolerant controller for θ.

Thus we have that:

Theorem 5. *Let $n \geq 1$ and let Σ be an alphabet. Let θ be a* CT-MITL *specification. Then we can construct a conflict-tolerant signal controller which meets the specification θ.*

9 Discussion

We have proposed a way of specifying conflict-tolerant real-time specifications in MITL. We believe this is a useful addition to the conflict-tolerant framework of [3, 4], as it is more natural for a user to specify properties in temporal logic. We use a clock-optimal construction for building a monitoring TTS for the $\diamondsuit_{\langle l,r\rangle}\psi$ inductive step, based on a "delay-then-extend" idea. We also use this idea to design simple controllers using standard "delay" and "extension" components.

References

[1] Rajeev Alur and David L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[2] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996.

[3] Deepak D'Souza and Madhu Gopinathan. Conflict-tolerant features. In Aarti Gupta and Sharad Malik, editors, *CAV*, volume 5123 of *Lecture Notes in Computer Science*, pages 227–239. Springer, 2008.

[4] Deepak D'Souza, Madhu Gopinathan, S. Ramesh, and Prahladavaradan Sampath. Conflict-tolerant real-time features. In *QEST*, pages 274–283. IEEE Computer Society, 2008.

[5] Madhu Gopinathan. *Conflict Tolerant Features*. PhD thesis, CSA Department, Indian Institute of Science, 2009.

[6] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[7] Oded Maler, Dejan Nickovic, and Amir Pnueli. Real time temporal logic: Past, present, future. In Paul Pettersson and Wang Yi, editors, *FORMATS*, volume 3829 of *Lecture Notes in Computer Science*, pages 2–16. Springer, 2005.

[8] Oded Maler, Dejan Nickovic, and Amir Pnueli. From mitl to timed automata. In Eugene Asarin and Patricia Bouyer, editors, *FORMATS*, volume 4202 of *Lecture Notes in Computer Science*, pages 274–289. Springer, 2006.

[9] Joseph Sifakis and Sergio Yovine. Compositional specification of timed systems (an extended abstract). In *Symp. on Theoretical Aspects of Comp. Science*, pages 347–359, 1996.

[10] Raj Mohan M. Sumesh Divakaran, Deepak D'Souza. Conflict-Tolerant Real-Time Specifications in Metric Temporal Logic. Technical Report IISc-CSA-TR-2009-9, CSA Department, Indian Institute of Science, 2009. URL: `http://archive.csa.iisc.ernet.in/TR/2009/9/`.

[11] Raj Mohan M. Sumesh Divakaran, Deepak D'Souza. Conflict-Tolerant Specifications in Temporal Logic. Technical Report IISc-CSA-TR-2009-8, CSA Department, Indian Institute of Science, 2009. URL: `http://archive.csa.iisc.ernet.in/TR/2009/8/`.