Mixed-Time Signal Temporal Logic

Thomas Ferrère[1], Oded Maler[2], Dejan Ničković[3]

[1] IST Austria, Klosterneuburg, Austria
[2] VERIMAG, University of Grenoble, France
[3] AIT Austrian Institute of Technology, Vienna, Austria

Abstract. We present Mixed-time Signal Temporal Logic (STL-MX), a specification formalism which extends STL by capturing the discrete/continuous time duality found in many cyber-physical systems (CPS), as well as mixed-signal electronic designs. In STL-MX, properties of components with continuous dynamics are expressed in STL, while specifications of components with discrete dynamics are written in LTL. To combine the two layers, we evaluate formulas on two traces, discrete- and continuous-time, and introduce two interface operators that map signals, properties and their satisfaction signals across the two time domains. We show that STL-MX has the expressive power of STL supplemented with an implicit T-periodic clock signal. We develop and implement an algorithm for monitoring STL-MX formulas and illustrate the approach using a mixed-signal example.

1 Introduction

Cyber-physical systems (CPS) typically combine together components with *continuous* dynamics (analog components, sensors, actuators) and components with *discrete* dynamics (digital controllers, software, firmware). Models of (most) components with discrete dynamics operate in discrete (clocked) time and manipulate values in a finite domain. In contrast, components with continuous dynamics are modeled as operating in continuous time over real-valued variables. The interaction between these two classes of heterogeneous components is typically done via *converters*, which allow passing from one time and value domain to another. For instance, analog and digital components in an analog mixed-signal (AMS) design can be integrated by inserting *analog-to-digital* (A/D) and *digital-to-analog* (D/A) converters providing the necessary interface, as illustrated in Figure 1.[4] The time and value domain differences between analog and digital components pose difficult design and verification challenges.

While correctness evidence is imposed for safety-critical CPS applications (see for example the automotive standard ISO 26262 [14]), CPS verification remains an important bottleneck in the development process, resulting in up to 70% of the project effort. Verification of CPS in industry is almost exclusively based on *simulation*, where each scenario can take several hours of simulation

[4] This is a simplification of the AMS setting: not all interaction between analog and digital components goes through A/D and D/A conversions.

time. Simulation traces are typically observed by verification engineers for correctness, resulting in a manual, ad-hoc and error-prone process.

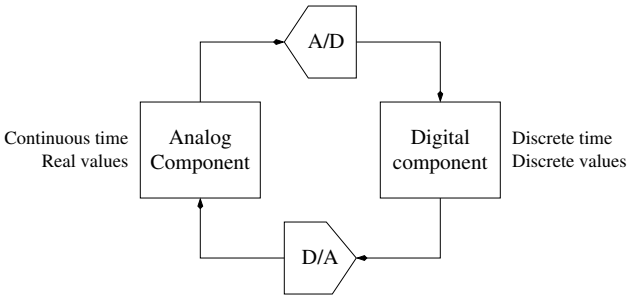

Fig. 1. A typical scenario in AMS design.

Specification-based monitoring is a pragmatic, yet rigorous approach for systematic simulation-based verification. *Signal Temporal Logic* (STL) [18,19] is a declarative specification language for describing properties of CPS behaviors. As an extension of the real-time temporal logics MTL [17] and MITL (*Metric Interval Temporal Logic* [2]), STL allows us to reason about real-time properties of real-valued signals. STL and its extensions have been used to specify and reason about properties of systems coming from Industry 4.0, semiconductor, automotive, avionics, medical devices and system and synthetic biology domains – see the survey [4] for a detailed list of references.

While STL effectively provides support for combining Boolean and real-valued signal properties, the time domain remains continuous for *all* signals. Such specifications are not fully aligned with the actual practice in development and integration of CPS where designers of discrete dynamics components often reason about time in terms of clock ticks, and hence the natural logic to express digital properties is a discrete-time temporal logic. It would be extremely counter-productive if verification engineers, due to few continuous/discrete dynamics interface properties, would have to transform all digital properties to dense time. We propose a simple and transparent solution in which both time models can co-exist. We illustrate the class of properties that motivate us via the following example.

Example 1. Consider the following stabilization property for a CPS system with sampling period $T = 200$. Whenever a discrete signal *cmd* is set up by the digital controller from false to true, the absolute value of a continuous signal x in an analog component must become lower than 1 within 600 time units and remain continuously within that range for at least 300 time units. This informal specification is illustrated in Figure 1.

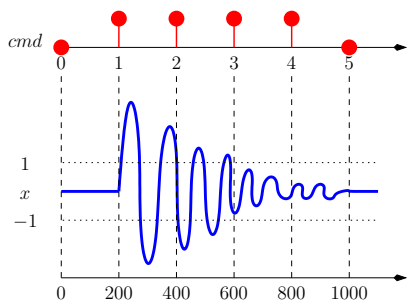

Fig. 2. Illustration of a stabilization specification.

We propose *Mixed-time Signal Temporal Logic* (STL-MX) as a specification language that extends STL to express properties both in terms of discrete logical time (clock ticks) and dense real time. In essence, STL-MX consists of two layers: (1) the standard discrete-time temporal logic LTL [22] for specification of digital component properties; and (2) STL for specification of analog component behaviors. To combine the two layers, we split the trace into a continuous-time and discrete-time part, and introduce two time-mapping operators $@^{\mathsf{cd}}$ and $@^{\mathsf{dc}}$ that formalize the conversions between continuous-time and discrete-time formulas and signals. We then study the expressiveness of this formalism and show that STL-MX can be effectively embedded into STL when provided with an explicit sampling signal. We present an implementation of the monitoring algorithm for STL-MX and demonstrate the utility of mixed-time specifications on a case study from the AMS domain.

Related Work The main inspiration for this paper comes from the work that introduces digital clocks into LTL [11]. That work does not consider continuous interpretation of time, in contrast to this paper.

In the past years, there has been a rich body of work on various extensions of MTL and STL. There, continuous-time specification languages were extended with various quantitative semantics [12,13,9,1,15]. In particular, STL was then extended with support for time-frequency properties [10] and freeze quantification [8]. A first-order logic of signals [3] has been recently developed as a generalization of STL and STL with freeze quantification. None of these extensions considers both discrete and continuous time interpretation of the logic at the same time.

The problem of different time domains has been also studied in other domains. Ptolemy [5] provides a prototyping and simulation environment for modelling heterogeneous systems that combine different models of computations. We also mention the GEMOC initiative [6] that is promoting coordinated use of modelling languages with possibly different models of computation, and hence time domains.

2 Mixed-Time Signal Temporal Logic

In this section we introduce the syntax of STL-MX and its semantics over both discrete-time and continuous-time signals. Since we are interested in monitoring we focus on signals of bounded duration. Without loss of generality we assume all digital signals in the circuit to range over the Booleans and analog signals to range over the reals.

Let $\mathbb{D}$ be a value domain, typically $\mathbb{B}$ or $\mathbb{R}$. A discrete-time signal w is a function $w : \{0, 1, \ldots, s\} \to \mathbb{D}$ for $s > 0$. A continuous-time signal u is a function $u : [0, r) \to \mathbb{D}$ for $r > 0$. We denote by $|w|$ and $|u|$ the *length* of signals w and u, respectively. In the rest of this paper, we use $r = |u|$ and $s = |w|$. Relative to u and v, a *sampling* $\tau : \{0, 1, \ldots, s\}$ is a monotonically increasing sequence of times $0 = \tau[0] < \tau[1] < \cdots < \tau[r] < s \in \mathbb{R}_{\geq 0}$. A sampling indicates the times at which the discrete values are read. We say that the sampling τ is *periodic* (with period $T \in \mathbb{R}_{>0}$) when $\tau[i] = iT$ for all $i \in \{0, 1, \ldots, s\}$.

A sequence of disjoint non-empty intervals $I_0 \cdot I_1 \cdots I_k$ is a time partition compatible with a finitely-varying continuous-time Boolean signal x if (1) $\bigcup_{0 \leq j \leq k} I_j = [0, |x|)$ and (2) Each I_j is of the form (t_j, t_{j+1}), $[t_j, t_{j+1})$, $(t_j, t_{j+1}]$ or $[t_j, t_{j+1}]$ such that $t_j \leq t_{j+1}$, $\forall t, t' \in I_j,\ x(t) = x(t')$. The *coarsest time partition* associated with x satisfies the additional property: (3) Whenever $t \in I_j$ and $t' \in I_{j+1}$ then $x(t) \neq x(t')$.

Let $P = \{p_1, \ldots, p_m\}$ be a set of Boolean variables and let $X = \{x_1, \ldots, x_n\}$ be a set of real valued variables. A constraint over X is a predicate of the form $x \prec c$, where $x \in X$, $\prec \in \{<, \leq, =, \geq, >\}$ and $c \in \mathbb{Q}$. A *mixed-time signal temporal logic* (STL-MX) formula ψ is either a continuous-time formula α or a discrete-time formula φ defined over X and P according to the following grammar[5]

$$\alpha ::= x \prec c \mid \neg\alpha \mid \alpha_1 \vee \alpha_2 \mid \alpha_1 \mathcal{U}_I \alpha_2 \mid \alpha_1 \mathcal{S}_I \alpha_2 \mid @^{\mathsf{dc}}(\varphi)$$
$$\varphi ::= p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \bigcirc \varphi \mid \ominus \varphi \mid \varphi_1 \mathcal{U} \varphi_2 \mid \varphi_1 \mathcal{S} \varphi_2 \mid @^{\mathsf{cd}}(\alpha)$$

where $\bigcirc$, $\ominus$, $\mathcal{U}$ and $\mathcal{S}$ are temporal *next*, *previous*, *until* and *since* operators, $p \in P$, $x \in X$, $c \in \mathbb{Q}$ and I is an interval of the form $[a, b]$, $[a, b)$, $(a, b]$, (a, b), $[a, \infty)$ or (a, ∞) where $0 \leq a < b$ are rational numbers.

We can define other usual operators $\Diamond_I$ (*eventually*), $\Box_I$ (*always*), $\boxminus_I$ (*once*), and $\boxminus_I$ (*historically*) as syntactic abbreviations with $\Diamond_I \varphi := \mathsf{true} \mathcal{U}_I \varphi$, $\Box_I \varphi := \neg \Diamond_I \neg\varphi$, and likewise for past operators – discrete time syntax can be enriched with corresponding constructs. Note that timing interval subscripts may be omitted in when equal to $[0, +\infty)$.

Example 2. We formalize the stabilization property from Example 1 by the following STL-MX formula:

$$\Box(((\ominus \neg cmd) \wedge cmd) \rightarrow @^{\mathsf{cd}}(\Diamond_{[0,600]} \Box_{[0,300]} (|x| < 1)))$$

[5] We use the same symbols for Boolean and temporal connectives in both continuous-time and discrete-time formulas. The distinction between the two layers is defined by the context. Note that each valid formula is classified unambiguously as discrete-time or continuous-time.

Let w be a discrete-time m-dimensional Boolean signal and u be a continuous-time real-valued n-dimensional signal, which we assume such that $u_\tau[i] \in [0, r)$ for all $i \in \{0, 1, \ldots, s\}$. We denote by $\pi_p(w)$ and $\pi_x(u)$ the respective projections of w and u on variables p and x. Conversely, for w and w' over the same time domain we denote by $w \| w'$ the pairing of signals w and w', such that $\pi_v(w \| w') = \pi_v(w)$ when variable v is a dimension of w and $\pi_v(w \| w') = \pi_v(w')$ when v is a dimension of w'. Finally, we use $\oplus$ for the Minkowski sum of intervals, that is, $[a_1, b_1] \oplus [a_2, b_2] = [a_1 + a_2, b_1 + b_2]$.

In what follows, we assume a sampling τ given independently and globally defined. The semantics of a discrete-time STL-MX formula φ with respect to signals w and u is described via the satisfaction relation $(w, u, i) \models^{\mathsf{d}} \varphi$, indicating that signals w and u satisfy φ at discrete time index i. Similarly, the semantics of a continuous-time STL-MX formula α with respect to signals w and u is described via the satisfaction relation $(w, u, t) \models^{\mathsf{c}} \alpha$, indicating that signals w and u satisfy α at time t. These relations are defined recursively below.

$$
\begin{aligned}
(w,u,i) &\models^{\mathsf{d}} p && \leftrightarrow \pi_p(w)[i] = 1 \\
(w,u,i) &\models^{\mathsf{d}} \neg\varphi && \leftrightarrow (w,u,i) \not\models^{\mathsf{d}} \varphi \\
(w,u,i) &\models^{\mathsf{d}} \varphi_1 \vee \varphi_2 && \leftrightarrow (w,u,i) \models^{\mathsf{d}} \varphi_1 \text{ or } (w,u,i) \models^{\mathsf{d}} \varphi_2 \\
(w,u,i) &\models^{\mathsf{d}} \bigcirc \varphi && \leftrightarrow i < |w| \text{ and } (w,u,i+1) \models^{\mathsf{d}} \varphi \\
(w,u,i) &\models^{\mathsf{d}} \ominus \varphi && \leftrightarrow i > 0 \text{ and } (w,u,i-1) \models^{\mathsf{d}} \varphi \\
(w,u,i) &\models^{\mathsf{d}} \varphi_1 \mathcal{U} \varphi_2 && \leftrightarrow \exists i \le i' < |w| \text{ s.t. } (w,u,i') \models^{\mathsf{d}} \varphi_2 \text{ and} \\
& && \quad \forall\, i \le i'' < i',\ (w,u,i'') \models^{\mathsf{d}} \varphi_1 \\
(w,u,i) &\models^{\mathsf{d}} \varphi_1 \mathcal{S} \varphi_2 && \leftrightarrow \exists 0 \le i' \le i \text{ s.t. } (w,u,i') \models^{\mathsf{d}} \varphi_2 \text{ and} \\
& && \quad \forall\, i' < i'' \le i,\ (w,u,i'') \models^{\mathsf{d}} \varphi_1 \\
(w,u,i) &\models^{\mathsf{d}} @^{\mathsf{cd}}(\alpha) && \leftrightarrow (w,u,\tau[i]) \models^{\mathsf{c}} \alpha
\end{aligned}
$$

$$
\begin{aligned}
(w,u,t) &\models^{\mathsf{c}} x \prec c && \leftrightarrow \pi_x(u)[t] \prec c \\
(w,u,t) &\models^{\mathsf{c}} \neg\alpha && \leftrightarrow (w,u,t) \not\models^{\mathsf{c}} \alpha \\
(w,u,t) &\models^{\mathsf{c}} \alpha_1 \vee \alpha_2 && \leftrightarrow (w,u,t) \models^{\mathsf{c}} \alpha_1 \text{ or } (w,u,t) \models^{\mathsf{c}} \alpha_2 \\
(w,u,t) &\models^{\mathsf{c}} \alpha_1 \mathcal{U}_I \alpha_2 && \leftrightarrow \exists t' \in (t \oplus I) \cap [0, |u|] \text{ s.t. } (w,u,t') \models^{\mathsf{c}} \alpha_2 \text{ and} \\
& && \quad \forall\, t'' \in (t, t'),\ (w,u,t'') \models^{\mathsf{c}} \alpha_1 \\
(w,u,t) &\models^{\mathsf{c}} \alpha_1 \mathcal{S}_I \alpha_2 && \leftrightarrow \exists t' \in (t \ominus I) \cap [0, |u|] \text{ s.t. } (w,u,t') \models^{\mathsf{c}} \alpha_2 \text{ and} \\
& && \quad \forall\, t'' \in (t', t),\ (w,u,t'') \models^{\mathsf{c}} \alpha_1 \\
(w,u,t) &\models^{\mathsf{c}} @^{\mathsf{dc}}(\varphi) && \leftrightarrow (w,u,\operatorname{argmax}_{i \in \{0,1,\ldots,s\}} \tau[i] \le t) \models^{\mathsf{d}} \varphi
\end{aligned}
$$

We use $(w, u) \models \psi$ as a shorthand for $(w, u, 0) \models^{\mathsf{d}} \psi$ or $(w, u, 0) \models^{\mathsf{c}} \psi$ according to the type of ψ. Based on these definitions and given a pair (w, v) we associate with each discrete-time formula φ and a continuous-time formula α their respective Boolean *satisfaction signals* w_φ and u_α such that for all $i \in \{0, 1, \ldots, s\}$ and $t \in [0, r)$, $w_\varphi[i] = 1$ iff $(w, u, i) \models^{\mathsf{d}} \varphi$ and $u_\alpha[t] = 1$ iff $(w, u, t) \models^{\mathsf{c}} \alpha$.

As we see, temporal operators inherit their semantics from LTL and STL. The semantics of new interface operators $@^{\mathsf{cd}}$ and $@^{\mathsf{dc}}$ are illustrated in Figure 3.

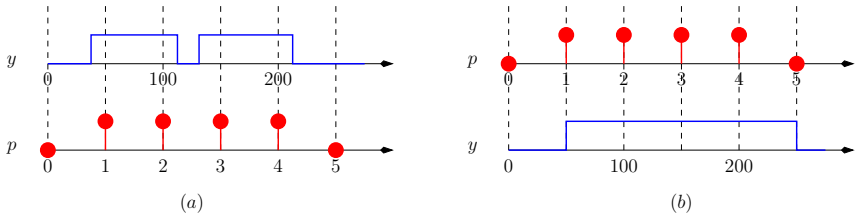

Fig. 3. Semantics of (a) $p = @^{\mathsf{cd}}(y)$ and (b) $y = @^{\mathsf{dc}}(p)$ for $T = 50$.

We remark that the choice of whether the top level STL-MX formula is continuous-time or discrete-time is typically application-dependent. Expressing a property of a discrete-time component (e.g. a controller) that drives another continuous-time component (e.g. a plant) usually results in a discrete-time top-level formula, and vice versa.

We now point out some properties of this logic. We say that two discrete-time formulas φ_1 and φ_2 are *equivalent*, denoted by $\varphi_1 \sim \varphi_2$ if for all signals u, w and time indexes i we have $(w, u, i) \models^{\mathsf{d}} \varphi_1 \leftrightarrow (w, u, i) \models^{\mathsf{d}} \varphi_2$. The equivalence between continuous-time formulas is defined similarly. It should be obvious that by converting a discrete-time formula into a continuous-time one, and in turn converting it back to a discrete-time formula we do not alter the meaning of the original formula. We illustrate this observation with the continuous-time signal (i.e. propositional formula) $y = @^{\mathsf{dc}}(p)$ from Figure 3 (b) obtained by converting the discrete-time signal p – it is clear that translating y back to a discrete-time signal yields p.

Proposition 1. *Any discrete-time formula φ satisfies $@^{cd}(@^{dc}(\varphi)) \sim \varphi$.*

This stems directly from the fact that any time index $i \in \{0, 1, \ldots, s\}$ verifies $\operatorname{argmax}_{j\in\{0,1,\ldots,n\}} \tau[j] \leq \tau[i] = i$. Conversely, for some time $t \in \mathbb{R}$ we generally do not have $\tau_{\operatorname{argmax}_{j\in\{0,1,\ldots,n\}}\tau[j]\leq t} = t$ hence nothing can be said of $@^{\mathsf{dc}}(@^{\mathsf{cd}}(\alpha))$ as compared with α, perhaps except for being a piecewise-constant approximation of its satisfaction signals.

Notwithstanding, time mapping operators commute with propositional connectives. For instance, first negating the propositional formula y from Figure 3 (a) and than coverting it to a discrete time formula is equivalent to first converting y to a discrete time formula and then doing the negation.

Proposition 2. *The following equivalences hold for any discrete-time formulas $\varphi, \varphi_1, \varphi_2$ and continuous-time formulas $\alpha, \alpha_1, \alpha_2$:*

$$@^{dc}(\neg\varphi) \sim \neg @^{dc}(\varphi) \qquad @^{dc}(\varphi_1 \vee \varphi_2) \sim @^{dc}(\varphi_1) \vee @^{dc}(\varphi_2)$$
$$@^{cd}(\neg\alpha) \sim \neg @^{cd}(\alpha) \qquad @^{cd}(\alpha_1 \vee \alpha_2) \sim @^{cd}(\alpha_1) \vee @^{cd}(\alpha_2)$$

Checking these facts is straightforward, let us for instance prove the first equivalence. Taking w, u some discrete and continuous signals and a time instant t,

we have $(w,u,t) \models^{\mathsf{c}} @^{\mathsf{dc}}(\neg\varphi) \leftrightarrow (w,u,\mathrm{argmax}_{j\in\{0,1,\ldots,n\}}\tau[j] \leq t) \models^{\mathsf{d}} \neg\varphi \leftrightarrow (w,u,\mathrm{argmax}_{j\in\{0,1,\ldots,n\}}\tau[j] \leq t) \not\models^{\mathsf{d}} \varphi \leftrightarrow (w,u,t) \not\models^{\mathsf{c}} @^{\mathsf{dc}}(\varphi) \leftrightarrow (w,u,t) \models^{\mathsf{c}} \neg @^{\mathsf{dc}}(\varphi)$.

As expected, temporal operators do not enjoy such properties. The following section may provide more insight in this respect.

3 Expressivity

STL-MX has similar expressive power as STL when supplemented by a "digital clock", as we show in the following.

Let w be a discrete signal. We say that a continuous signal w^τ is the *right-continuation* of w when $w^\tau[t] = w[\mathrm{argmax}_{j\in\{0,1,\ldots,n\}}\tau[j] \leq t]$ for all $t \in [0,r)$. Note that on discrete Boolean signals, the interpretation of $@^{\mathsf{dc}}$ is exactly right-continuation with period T. Conversely, the interpretation of $@^{\mathsf{cd}}$ is the sampling of continuous signals at *absolute* times $\tau[i]$, $i \in \{0,1,\ldots,s\}$. For this purpose, let us introduce a special continuous Boolean signal clk with the following definition:

$$\mathsf{clk} : t \mapsto \begin{cases} 1 & \text{when } t = \tau[i] \text{ for some } i \in \{0,1,\ldots,s\} \\ 0 & \text{otherwise} \end{cases}$$

Following definitions of Section 2, let us define STL to be continuous-time STL-MX formulas without discrete-time sub-formulas. For a continuous signal u, a time t, and an STL formula α the standard STL semantics reads $(\emptyset,u,t) \models^{\mathsf{c}} \alpha$.

We now inductively define a syntactical mapping σ from STL-MX to STL formulas:

$$\begin{aligned}
\sigma(p) &= p & \sigma(@^{\mathsf{cd}}(\alpha)) &= \neg\mathsf{clk}\,\tilde{\mathcal{S}}(\mathsf{clk} \wedge \sigma(\alpha)) \\
\sigma(\bigcirc\varphi) &= \neg\mathsf{clk}\,\mathcal{U}(\mathsf{clk} \wedge \sigma(\varphi)) & \sigma(\neg\varphi) &= \neg\sigma(\varphi) \\
\sigma(\ominus\varphi) &= \neg\mathsf{clk}\,\mathcal{S}(\mathsf{clk} \wedge \sigma(\varphi)) & \sigma(\varphi_1 \vee \varphi_2) &= \sigma(\varphi_1) \vee \sigma(\varphi_2) \\
\sigma(\varphi_1\,\mathcal{U}\,\varphi_2) &= \sigma(\varphi_2) \vee (\sigma(\varphi_1)\,\mathcal{U}_{(0,+\infty)}\,\sigma(\varphi_2)) \\
\sigma(\varphi_1\,\mathcal{S}\,\varphi_2) &= \sigma(\varphi_2) \vee (\sigma(\varphi_1) \wedge \sigma(\varphi_1)\,\mathcal{S}_{(0,+\infty)}(\sigma(\varphi_2)\,\mathcal{S}_{(0,+\infty)}\mathsf{true}))
\end{aligned}$$

$$\begin{aligned}
\sigma(x \prec c) &= x \prec c & \sigma(@^{\mathsf{dc}}(\varphi)) &= \sigma(\varphi) \\
\sigma(\alpha_1\,\mathcal{U}_I\,\alpha_2) &= \sigma(\alpha_1)\,\mathcal{U}_I\,\sigma(\alpha_2) & \sigma(\neg\alpha) &= \neg\sigma(\alpha) \\
\sigma(\alpha_1\,\mathcal{S}_I\,\alpha_2) &= \sigma(\alpha_1)\,\mathcal{S}_I\,\sigma(\alpha_2) & \sigma(\alpha_1 \vee \alpha_2) &= \sigma(\alpha_1) \vee \sigma(\alpha_2)
\end{aligned}$$

This mapping is such that an STL-MX formula ψ is satisfied by some signal if and only if its STL translation is satisfied by the right-continuation of the discrete signal, paired with the original continuous signal and the clock. Note that in STL p is associated with a continuous-time satisfaction signal.

Example 3. The stabilization property (2) is mapped into the following STL formula.

$$\square((\neg\mathsf{clk}\,\mathcal{S}(\mathsf{clk} \wedge \neg cmd)) \wedge cmd) \rightarrow (\neg\mathsf{clk}\,\tilde{\mathcal{S}}(\mathsf{clk} \wedge \diamondsuit_{[0,600]}\,\square_{[0,300]}\,|x<1|))$$

Theorem 1. *Let w be a discrete-time signal and let u be a continuous-time signal. Taking w^τ as the right-continuation of w, we have:*

1. *for any discrete-time* STL-MX *formula φ and $i \in \mathbb{N}$*

$$(w, u, i) \models^{\mathsf{d}} \varphi \quad \textit{iff} \quad (\emptyset, w^\tau \| u \| \mathsf{clk}, \tau[i]) \models^{\mathsf{c}} \sigma(\varphi)$$

2. *for any continuous-time* STL-MX *formula α and $t \in \mathbb{R}_{\geq 0}$*

$$(w, u, t) \models^{\mathsf{c}} \alpha \quad \textit{iff} \quad (\emptyset, w^\tau \| u \| \mathsf{clk}, t) \models^{\mathsf{c}} \sigma(\alpha)$$

Proof. (1) and (2) are shown conjointly by induction on the formula structure. For propositions in P we have $\pi_p(w^\tau)[\tau[i]] = \pi_p(w)[i]$ from the definition of w^τ. Boolean connectives naturally commute with the right-continuation operation as seen in Proposition 2. Now looking at the $\bigcirc$ operator, we can check that for a discrete-time formula φ and a discrete signal w, the right-continuation of $w_{\bigcirc \varphi}$ requires that w_φ holds at the previous discrete time value. Operator $\ominus$ is symmetrical. The discrete *until* does not pose any problem. For the discrete *since*, note that its continuous counterpart $\mathcal{S}_{(0,\infty)}$ has *left*-continuous semantics. Notably, rather than looking for a witness of φ_2 at some time t' in the past, we look for a time t' such that φ_2 holds immediately before; this is done by the formula $\sigma(\varphi_2)\,\mathcal{S}_{(0,+\infty)}\mathsf{true}$. Concerning operator $@^{\mathsf{dc}}$, the translation states the continuous formula α was true on the previous clock tick. For pure continuous-time operators, semantics are unchanged so we only apply recursively the translation to treat possible discrete sub-formulas. Finally, the $@^{\mathsf{cd}}$ operator performs a right-continuation operation, hence we only need to translates discrete-time formulas it applies to.

Corollary 1. *For a periodic sampling $\tau[i] = iT$, the satisfaction of an* STL-MX *formula by a signal reduces (in polynomial time) to the satisfaction of an* STL *formula.*

Proof. A T-periodic clock is definable in STL by the formula

$$\delta_T := \mathsf{clk} \wedge \square(\mathsf{clk} \rightarrow (\square_{(0,T)}\, \neg\mathsf{clk} \wedge \diamondsuit_{(0,T]}\, \mathsf{clk}))$$

Assuming such a clock signal, we may impose on any Boolean signal p to be a T-period right-continuation signal using the formula

$$\gamma_p := \square \begin{pmatrix} ((\neg p\, \mathcal{S}_{(0,+\infty)}\mathsf{true} \wedge p) \rightarrow \mathsf{clk}) \ \wedge \ ((p\, \mathcal{S}_{(0,+\infty)}\mathsf{true} \wedge \neg p) \rightarrow \mathsf{clk}) \\ \wedge\ (p \rightarrow p\mathcal{U}_{(0,+\infty)}\mathsf{true}) \ \wedge \ (\neg p \rightarrow \neg p\mathcal{U}_{(0,+\infty)}\mathsf{true}) \end{pmatrix}$$

The two first conjuncts ensure that left-discontinuities can only occur on clock ticks, while the last two conjuncts enforce right-continuity. Now given some STL-MX formula ψ, we construct the STL formula

$$\psi' := \sigma(\psi) \wedge \delta_T \wedge \bigwedge_{p \in P} \gamma_p$$

It follows from Theorem 1 that ψ is satisfied if and only if ψ' is satisfied. Clearly the size of ψ' is linear in the size of ψ.

4 Monitoring STL-MX

In this section, we present the monitoring procedure for STL-MX. Like previous work on STL monitoring [21], our procedure is closely related to the idea of *temporal testers* advocated in [16,24] for discrete time and dating, in fact, back to [23]. These are *acausal* transducers that realize the semantics of the temporal logic operators as follows. For an operator OP interpreted over time domain $\mathbb{T}$, the temporal tester $\mathcal{T}_{\mathsf{OP}}$ takes as input a Boolean $\mathbb{T}$-signal y and outputs another Boolean signal y' such that if y is the satisfaction signal of some formula φ then y' is the satisfaction signal of $\mathsf{OP}\varphi$. For example the tester for *next* in discrete time realizes a forward shift, that is, $y'[i] = y[i+1]$. A temporal tester for a compound temporal logic formula φ is obtained by composing temporal testers for the basic operators, following the parse tree of the formula.

Testers have been proposed for several temporal logics, in particular for LTL [24] and MITL [20]. The monitoring procedure for STL described in [19] follows closely the temporal testers paradigm although the transduction function is computed directly on signals, without explicit construction of automata as in [20]. The compositional structure of temporal testers allows us to fully separate the monitoring of the LTL and STL components of an STL-MX formula and reuse existing results. We focus in this paper on the construction of temporal testers for the two additional operators $@^{\mathsf{cd}}$ and $@^{\mathsf{dc}}$ that interface discrete and continuous time. We refer the reader to [24,20,19] for details regarding the other operators. A high-level overview of the procedure as applied to the stabilization formula (2) is provided in Figure 4.

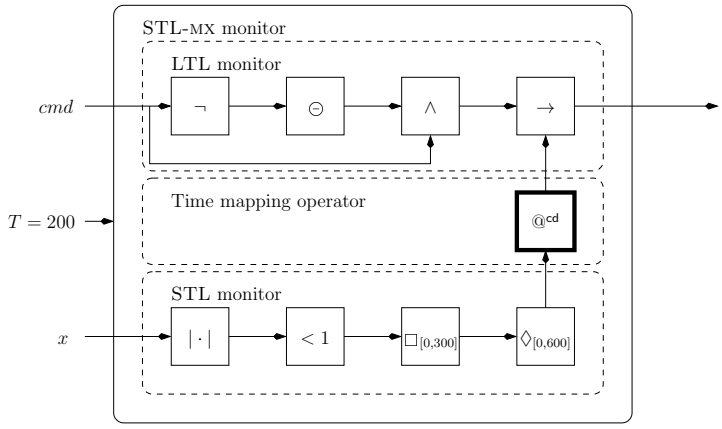

Fig. 4. Monitor based on temporal testers for formula (2).

The temporal tester $\mathcal{T}_{@^{\mathsf{cd}}}$, realizing the semantics of the $@^{\mathsf{cd}}$ operator, takes as input a continuous-time Boolean signal u, and outputs a discrete-time Boolean

signal w obtained by sampling u at multiples of T. The computation, based on a time partition $I_0 \cdot I_1 \cdots I_n$ compatible with u, is illustrated in Algorithm 1.

Algorithm 1 Temporal tester $\mathcal{T}_{@^{\mathsf{cd}}}$.

Require: Continuous Boolean signal u, *a* sampling period T
Ensure: Discrete Boolean signal $w = @^{\mathsf{cd}}(u)$
1: $k \leftarrow 0$
2: **for** $j = 0$ to n **do**
3: **while** $\tau[k] \in I_j$ **do**
4: $w[k] \leftarrow u(I_j)$
5: $k \leftarrow k + 1$
6: **end while**
7: **end for**
8: **return** w

Proposition 3. *Given a satisfaction signal y of a continuous-time formula α, a time index i, we have that $\mathcal{T}_{@^{cd}}(y)[i] = 1$ if and only if the satisfaction signal $u_{@^{cd}(\alpha)}[i] = 1$.*

The temporal tester $\mathcal{T}_{@^{\mathsf{dc}}}$, realizing the semantics of the $@^{\mathsf{dc}}$ operator, takes as input a discrete Boolean signal w and outputs a continuous-time Boolean signal u which "extends" the value of w at every time index i by holding it throughout the interval $[\tau[i], \tau[i+1])$. The procedure for computing $\mathcal{T}_{@^{\mathsf{dc}}}$ is illustrated in Algorithm 2. Note that the time partition created by the procedure is not the coarsest one compatible with u and it can be minimized later for efficiency.

Algorithm 2 Temporal tester $\mathcal{T}_{@^{\mathsf{dc}}}$.

Require: Discrete Boolean signal w, sampling period T
Ensure: Continuous Boolean signal u, the right continuation of w
1: $k \leftarrow 0$
2: **for** $j = 0$ to $|w|$ **do**
3: $I_k \leftarrow [\tau[j], \tau[j+1])$
4: $u[I_k] \leftarrow w[j]$
5: **end for**
6: **return** u

Proposition 4. *Given a discrete-time Boolean signal p, a time t and a sampling period T, we have that $\mathcal{T}_{@^{dc}}(p)[t] = 1$ if and only if the satisfaction signal $w_{@^{dc}(p)}[t] = 1$.*

For every other operator OP in the syntax, we already dispose of a transducer $\mathcal{T}_{\mathsf{OP}}$ with the corresponding lemma. We can now state the main result that our monitoring procedure computes the appropriate satisfaction signals.

Theorem 2. *Let w and u be discrete-time and continuous time-signals and let T be a sampling period. Then*

1. *For a discrete-time* STL-MX *formula φ, $\mathcal{T}_\varphi(w,u) = w_\varphi$;*
2. *For a continuous-time* STL-MX *formula α, $\mathcal{T}_\alpha(w,u) = u_\alpha$.*

We implemented the monitoring procedure for STL-MX formulas, following the structure shown in Figure 4 and applying STL-MX operations directly to discrete-time and continuous-time signals. The implementation consists of three layers: an LTL monitor, an STL monitor and the time mapping operations. Keeping the separation between the three layers allows us to monitor not only STL-MX specifications, but also its LTL and STL subsets for purely digital and analog applications, respectively. The implementation was written in C++ for GNU/Debian Linux x86 machines.

5 Case Study

We applied the monitoring implementation for STL-MX to verify basic properties of a simplified model of a Δ-Σ modulator, a basic component in analog to digital conversion. The circuit has an analog input, and a clocked digital output for typical integration in an ADC circuit. It is composed of the following building blocks: subtractor, integrator, threshold, and pulse generator. The overall architecture appears at Figure 5.

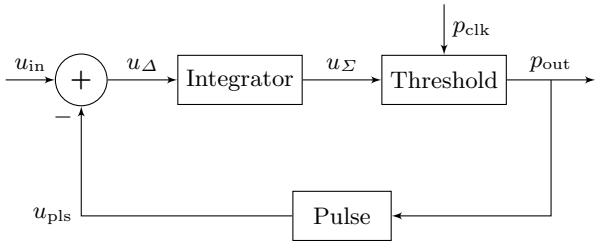

Fig. 5. Block diagram for the Δ-Σ modulator

The input voltage is first summed with the negated output of the control loop. The resulting voltage is integrated over time; when a value superior to a constant v_0 is reached, a threshold crossing is detected and we see a rising edge in the output. This signal is used to generate a pulse which is subtracted from the input, closing the loop. The effect is that during this pulse the integral sharply goes back below the threshold, and the cycle goes on. In addition, a clock is introduced so as to facilitate synchronization of the digital output. It is placed at the threshold detection level; rather than precisely detecting a crossing we

simply test for crossings on clock edges. Here is a short mathematical description of the idealized components realizing this behavior:

$$u_{\Delta}[t] = u_{\text{in}}[t] - u_{\text{pls}}[t] \qquad \text{(substractor)}$$

$$u_{\Sigma}[t] = A \cdot \int_0^t u_{\Delta}[t']\,\mathrm{d}t' \qquad \text{(integrator)}$$

$$p_{\text{out}}[i] = \begin{cases} 1 \text{ if } u_{\Sigma}[i\,T] \geq v_0 \\ 0 \text{ otherwise} \end{cases} \qquad \text{(threshold)}$$

$$u_{\text{pls}}[t] = \begin{cases} v_1 & \text{if } p_{\text{out}}[\lfloor \frac{t}{T} \rfloor - 1] = 0 \text{ and } p_{\text{out}}[\lfloor \frac{t}{T} \rfloor] = 1 \\ & \text{and } t - \lfloor \frac{t}{T} \rfloor \cdot T \leq T_{\text{pls}} \\ v_0 & \text{otherwise} \end{cases} \qquad \text{(pulse)}$$

where $T = 3.2\mu\text{s}$ is the period. We can see that in our model, the output is clocked with a frequency of 312 500Hz. The integrator gain is set to $A = 10^5$, the voltage threshold is $v_0 = 0.0\text{V}$. The pulse generator outputs piecewise constant signals, with high voltage of $v_1 = 3.3\text{V}$ and hold time of $T_{\text{pls}} = 2.5\mu\text{s}$. We have implemented the circuit as a mixed-signal model using Mentor Graphics' Questa ADMS [7] and simulated it against a variety of input signals. The simulation traces thus generated have been monitored with respect to STL-MX properties by our implementation of the procedure described in this paper.

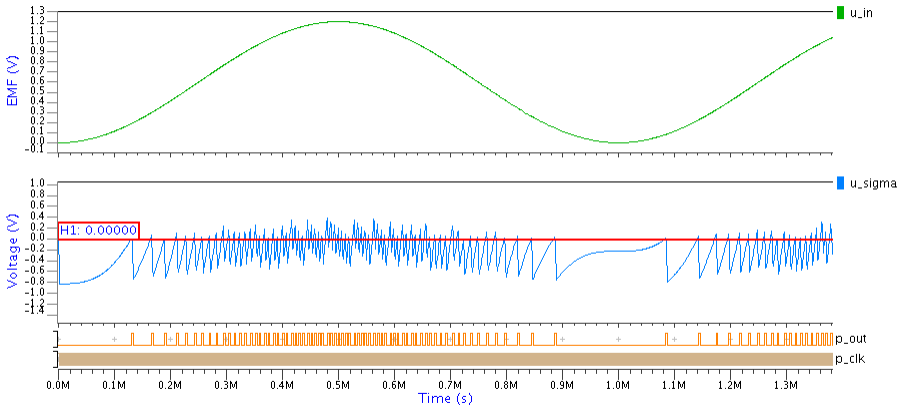

Fig. 6. Simulation trace (w_1, u_1) extracted with $u_{\text{in}} : t \mapsto 0.6\cos(1000 \cdot 2\pi \cdot t) + 0.6$.

We start with the following safety property:

Property 1. When we observe a rise in the output, the voltage out of the integrator has to return to a value below the threshold at the next clock tick.

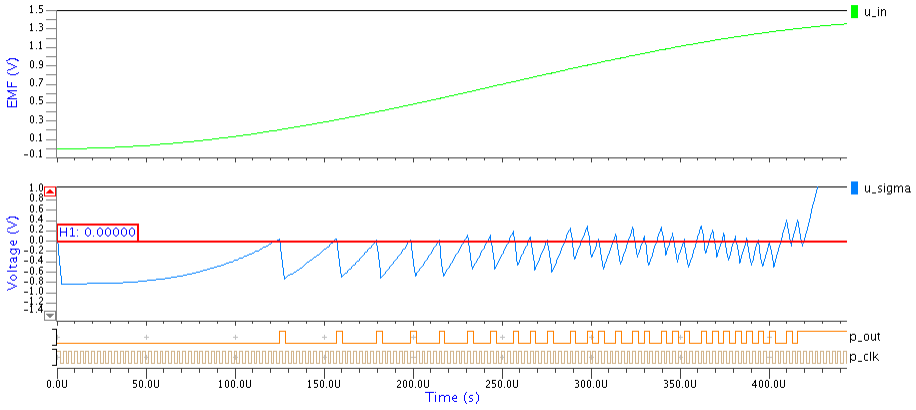

Fig. 7. Simulation trace (w_2, u_2) extract with $u_{\text{in}} : t \mapsto 0.7\ \cos(1000 \cdot 2\pi \cdot t) + 0.7$

This property is expressed in STL-MX as:

$$\psi_1 := \square((\ominus\, \neg p_{\text{out}} \wedge p_{\text{out}}) \rightarrow \bigcirc\, @^{\text{cd}} u_\Sigma < v_0)$$

The specification ψ_1 would be used during the integration steps of the design cycle so as to check that the input applied to the Δ-Σ modulator has a range conforming to its sampling capabilities.
First we simulate the design with a sinusoidal input at rate 1kHz and amplitude 0.6V; this gives us the trace (w_1, u_1) of Fig. 5. The circuit appears to behave adequately, and we have $(w_1, u_1) \models \psi_1$. When we modify slightly the input by setting the amplitude to 0.7V; the simulation produces the trace (w_2, u_2) of Fig. 5. We can detect a failure of our second property around 420μs, as the signal u_Σ goes back above v_0 within a single clock period. In our implementation the signal u_Σ would then indefinitely stay above the threshold, stalling the modulation. The algorithm that concludes that $(u_2, w_2) \not\models \psi_1$.

The Δ-Σ modulator should also verify some some functional specifications, for instance:

Property 2. When the input voltage is above 1.05V for 12.8μs the output must have a sequence of two consecutive spikes starting over that time frame.

Such a property, which can be used during the design phase of the Δ-Σ modulator itself, ia expressed as the following STL-MX formula:

$$\psi_2 := \square(\square_{[0,12.8]}\, u_{\text{in}} > 1.05 \rightarrow \Diamond_{[0,12.8]}\, @^{\text{dc}}(\neg p_{\text{out}} \wedge \bigcirc\, p_{\text{out}} \wedge \bigcirc^2 \neg p_{\text{out}} \wedge \bigcirc^3 p_{\text{out}}))$$

We test this specification on our design for several inputs of the form $t \mapsto A_1\ \cos(f_1 \cdot 2\pi \cdot t) + A_2\ \cos(f_2 \cdot 2\pi \cdot t) + B$ with $A_1 + A_2 + B = 1.2$ and f_1, f_2

ranging from 500Hz to 10kHz. The property is satisfied as long as the frequency in the input stays small; on the other hand rapidly varying signals introduce quantization uncertainty, and the property no longer holds. In all 6 simulation scenarios, we were able to show that ψ_2 is satisfied.

Table 1. STL-MX monitoring execution times.

Property	Sim. nb.	u_Σ	u_{in}	p_{out}	time (ms)
ψ_1	1	20 470		727	143
ψ_1	2	2 771		58	104
ψ_2	3		26 207	971	45
ψ_2	4		27 926	971	50
ψ_2	5		29 495	971	51
ψ_2	6		31 298	1 212	58
ψ_2	7		32 133	1 212	59
ψ_2	8		33 005	1 212	61

In Table 1, we present the evaluation of the STL-MX implementation to the Δ-Σ modulator case study. The experiments were done on an Intel Core i7-2620M CPU @ 2.7GHz machine with 8GB of RAM with the Windows 7 Enterprise operating system. The implementation was executed on Ubuntu 13.04 Linux operating system running on the Windows VMware Player 5.0.2 virtual machine. The table shows for a given STL-MX property, the size of the input signals in terms of the number of samples and the execution for monitoring the property, measured in milliseconds. The evaluation results show that the monitoring procedure induces minimal overhead, since for both properties ψ_1 and ψ_2, the time needed to monitor input of size ranging between $21,000$ and $35,000$ samples never exceeded 150 milliseconds.

Finally, we compare the STL-MX specification ψ_2 to the STL specification $\psi_2' = \sigma(\psi_2)$, where

$$\begin{aligned}\psi_2' := \square(\square_{[0,12.8]}\, u_{\text{in}} > 1.05 \rightarrow \diamondsuit_{[0,12.8]}(\neg p_{\text{out}} \wedge (\neg \mathsf{clk}\,\mathcal{U}(\mathsf{clk} \wedge p_{\text{out}})) \wedge \\ (\neg \mathsf{clk}\,\mathcal{U}\mathsf{clk} \wedge (\neg \mathsf{clk}\,\mathcal{U}(\mathsf{clk} \wedge \neg p_{\text{out}}))) \wedge \\ (\neg \mathsf{clk}\,\mathcal{U}\mathsf{clk} \wedge (\neg \mathsf{clk}\,\mathcal{U}(\mathsf{clk} \wedge (\neg \mathsf{clk}\,\mathcal{U}(\mathsf{clk} \wedge p_{\text{out}})))))\end{aligned}$$

This example demonstrates the potential value of explicitely separating the two time domains in specifications, which results in formulas that are more succint and easier to read.

6 Concluding Remarks

We have introduced very useful syntactic and semantic constructs that provide for co-existence of discrete and continuous-time specifications for runtime monitoring of CPS and mixed signal designs. This work is a first step toward a framework for system-wide specification-based verification, covering both discrete-time, bounded-value and continuous-time, real-valued domains. We studied the

theoretical properties of this mixed-time logic STL-MX and extended a monitoring framework to handle these two time domains. We demonstrated the usability of the methodology and tool on a case-study. As for the future one may think of the following directions:

1. Automatic insertion of $@^{\mathsf{cd}}$ and $@^{\mathsf{dc}}$ conversion operators based on type inference so as to facilitate further the expression of properties by the user;
2. Studying other conversion operators, more sophisticated than the currently used periodic sample and hold. For example, the truth value of a discrete-time signal at i can be based on integrating values at continuous time in some interval around iT. One can also think of event-based conversion in asynchronous style, unlike this work that focused on clocked digital components;
3. Studying a tighter interaction between the monitoring procedure and the simulators that generate the heterogeneous traces.
4. Equpping STL-MX with quantitative semantics. We expect that adding quantitative semantics based on the infinity norm to STL-MX shall be straightforward. However, we will need to investigate whether the basic properties of the language would be still preserved under this quantitative semantics. In addition, it would be an interesting challenge to add a more cumulative or average-based semantics to the specification language.

Acknowledgments This research was supported in part by the Austrian Science Fund (FWF) under grants 27 S11402-N23 (RiSE/SHiNE) and Z211-N23 (Wittgenstein Award), and by the Productive 4.0 project (ECSEL 737459). The ECSEL Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Denmark, Germany, Finland, Czech Republic, Italy, Spain, Portugal, Poland, Ireland, Belgium, France, Netherlands, United Kingdom, Slovakia, Norway.

References

1. Akazaki, T., Hasuo, I.: Time robustness in MTL and expressivity in hybrid system falsification. In: Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part II. pp. 356–374 (2015). https://doi.org/10.1007/978-3-319-21668-3_21
2. Alur, R., Feder, T., Henzinger, T.: The benefits of relaxing punctuality. Journal of the ACM **43**(1), 116–146 (1996). https://doi.org/http://doi.acm.org/10.1145/227595.227602
3. Bakhirkin, A., Ferrère, T., Henzinger, T.A., Nickovic, D.: The first-order logic of signals: keynote. In: Proceedings of the International Conference on Embedded Software, EMSOFT 2018, Torino, Italy, September 30 - October 5, 2018. p. 1 (2018). https://doi.org/10.1109/EMSOFT.2018.8537203
4. Bartocci, E., Deshmukh, J.V., Donzé, A., Fainekos, G.E., Maler, O., Nickovic, D., Sankaranarayanan, S.: Specification-based monitoring of cyber-physical systems: A survey on theory, tools and applications. In: Lectures on Runtime Verification - Introductory and Advanced Topics, pp. 135–175 (2018). https://doi.org/10.1007/978-3-319-75632-5_5

5. Buck, J.T., Ha, S., Lee, E.A., Messerschmitt, D.G.: Ptolemy: A framework for simulating and prototyping heterogenous systems. Int. Journal in Computer Simulation **4**(2) (1994)
6. Combemale, B., DeAntoni, J., France, R.B., Barn, B., Clark, T., Frank, U., Kulkarni, V., Turk, D. (eds.): Joint Proceedings of the First International Workshop On the Globalization of Modeling Languages (GEMOC 2013) and the First International Workshop: Towards the Model Driven Organization (AMINO 2013) Co-located with the 16th International Conference on Model Driven Engineering Languages and Systems (MODELS 2013), Miami, USA, September 29 - October 04, 2013, CEUR Workshop Proceedings, vol. 1102. CEUR-WS.org (2013), `http://ceur-ws.org/Vol-1102`
7. Graphics Corporation, M.: Questa ADMS. `http://www.mentor.com/products/fv/advance_ms/`
8. Dluhos, P., Brim, L., Safránek, D.: On expressing and monitoring oscillatory dynamics. In: HSB. pp. 73–87 (2012). https://doi.org/10.4204/EPTCS.92.6
9. Donzé, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In: Formal Modeling and Analysis of Timed Systems (FORMATS). pp. 92–106 (2010). https://doi.org/10.1007/978-3-642-15297-9_9
10. Donzé, A., Maler, O., Bartocci, E., Nickovic, D., Grosu, R., Smolka, S.A.: On temporal logic and signal processing. In: ATVA. pp. 92–106 (2012). https://doi.org/10.1007/978-3-642-33386-6_9
11. Eisner, C., Fisman, D., Havlicek, J., McIsaac, A., Campenhout, D.V.: The definition of a temporal clock operator. In: ICALP. pp. 857–870 (2003). https://doi.org/10.1007/3-540-45061-0_67
12. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications. In: Formal Approaches to Software Testing and Runtime Verification, First Combined International Workshops, FATES 2006 and RV 2006, Seattle, WA, USA, August 15-16, 2006, Revised Selected Papers. pp. 178–192 (2006). https://doi.org/10.1007/11940197_12
13. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications for continuous-time signals. Theor. Comput. Sci. **410**(42), 4262–4291 (2009). https://doi.org/10.1016/j.tcs.2009.06.021
14. ISO 26262:2011: Road Vehicles – Functional Safety. ISO, Geneva, Switzerland
15. Jaksic, S., Bartocci, E., Grosu, R., Nickovic, D.: Quantitative monitoring of STL with edit distance. In: Runtime Verification - 16th International Conference, RV 2016, Madrid, Spain, September 23-30, 2016, Proceedings. pp. 201–218 (2016). https://doi.org/10.1007/978-3-319-46982-9_13
16. Kesten, Y., Pnueli, A.: A compositional approach to CTL* verification. Theor. Comput. Sci. **331**(2-3), 397–428 (2005). https://doi.org/10.1016/j.tcs.2004.09.023
17. Koymans, R.: Specifying real-time properties with metric temporal logic. Real-time systems **2**(4), 255–299 (1990). https://doi.org/10.1007/BF01995674
18. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: FORMATS/FTRTFT. pp. 152–166 (2004). https://doi.org/10.1007/978-3-540-30206-3_12
19. Maler, O., Nickovic, D.: Monitoring properties of analog and mixed-signal circuits. STTT **15**(3), 247–268 (2013). https://doi.org/10.1007/s10009-012-0247-9
20. Maler, O., Nickovic, D., Pnueli, A.: From MITL to timed automata. In: Formal Modeling and Analysis of Timed Systems. pp. 274–289. Springer (2006). https://doi.org/10.1007/11867340_20

21. Maler, O., Nickovic, D., Pnueli, A.: Checking temporal properties of discrete, timed and continuous behaviors. In: Pillars of Computer Science. pp. 475–505 (2008). https://doi.org/10.1007/978-3-540-78127-1_26
22. Manna, Z., Pnueli, A.: Temporal Logic. Springer (1992)
23. Michel, M.: Computation of temporal operators. Logique et Analyse **110-111**, 137–152 (1985)
24. Pnueli, A., Zaks, A.: On the merits of temporal testers. In: 25 Years of Model Checking. pp. 172–195 (2008). https://doi.org/10.1007/978-3-540-69850-0_11