Robust Satisfaction of Temporal Logic over Real-Valued Signals

Conference Paper · September 2010

DOI: 10.1007/978-3-642-15297-9_9 · Source: DBLP

2 authors:

Alexandre Donzé
University of California, Berkeley
83 PUBLICATIONS **2,779** CITATIONS

SEE PROFILE

Oded Maler
French National Centre for Scientific Research
235 PUBLICATIONS **8,988** CITATIONS

SEE PROFILE

Robust Satisfaction of Temporal Logic over Real-Valued Signals

Alexandre Donzé and Oded Maler

CNRS-Verimag, 2 Av. de Vignate, 38610 Gières, France
@imag.fr

Abstract. We consider temporal logic formulae specifying constraints in continuous time and space on the behaviors of continuous and hybrid dynamical system admitting uncertain parameters. We present several variants of robustness measures that indicate how far a given trajectory stands, in space and time, from satisfying or violating a property. We present a method to compute these robustness measures as well as their sensitivity to the parameters of the system or parameters appearing in the formula. Combined with an appropriate strategy for exploring the parameter space, this technique can be used to guide simulation-based verification of complex nonlinear and hybrid systems against temporal properties. Our methodology can be used for other non-traditional applications of temporal logic such as characterizing subsets of the parameter space for which a system is guaranteed to satisfy a formula with a desired robustness degree.

1 Introduction

Analyzing the behavior of complex hybrid and nonlinear systems admitting uncertain parameters and inputs is an important ingredient in the design of *control systems* and *analog circuits* as well as in studying *biochemical reactions*. The adaptation of verification techniques to this domain proceeds along different threads, two of which are combined in the present paper. *Property checking*, also known as *monitoring* or *runtime verification*, uses temporal formulae to express desired behaviors and then checks whether individual system behaviors satisfy them, without worrying whether the set of behaviors checked *covers* the reachable state space. *Simulation-based verification* attempts to guide the generation of traces so as to demonstrate the satisfaction or violation of a property based on finitely many of them. In [8] we developed an algorithm that verifies this way safety properties of high-dimensional nonlinear systems. In this work we extend this technique to arbitrary properties expressed in a suitable temporal logic, for which we have devised a monitoring procedure [15]. The approach of [8] is strongly based on *sensitivity*, which in a nutshell, is the derivative of one continuous quantity with respect to another. To apply this concept to temporal formulae, we need to "continualize" their semantics by making it real-valued, to extend the monitoring procedure to compute this semantics efficiently for a given trace and compute the sensitivity of this semantics to parameters. This is what we do in the present paper after giving some background and motivation.

The introduction by Amir Pnueli [20] of linear-time temporal logic (LTL) into systems design as a formalism for specifying desirable and acceptable behaviors of reactive

systems is recognized as an important turning point in verification, putting the *ongoing sequential behavior* of a system at the center stage of the verification process. The verification framework developed over the years by Manna and Pnueli [18] consisted of three major components:

1. A *system description formalism*, specifying a behavior-generating mechanism S;
2. A *property specification formalism*, describing sets of acceptable behaviors, those that satisfy a formula φ;
3. A *verification methodology* for checking whether all behaviors of S satisfy φ.

In addition to verification where one checks whether *all* system behaviors satisfy φ, LTL is also used for lightweight verification (monitoring, runtime verification [5]) where the satisfaction of a formula by one or more *individual* behaviors is checked. For both uses, monitoring and verification alike, we point out three fundamental features of this framework, the first two related to the nature of the systems and behaviors considered while the third is related to the very notion of property and logical truth.

1. *Discrete qualitative time*: behaviors are defined as *sequences* of states or events, which are often interpreted in a purely qualitative manner, that is, without considering the metric distance between subsequent time instances;
2. *Discrete state space*: the sequences are defined over discrete and often finite domains emphasizing control rather than data;
3. *Yes/No answer*: the satisfaction of a temporal formula by a behavior is considered as membership in a set, with no quantitative degree of satisfaction.[1]

The first two features are natural for all sorts of *transition system* models expressed in various syntactic forms. Logically speaking, they imply the use of *discrete-time* and typically *propositional* temporal logic. The third feature often goes without saying. In the past two decades various attempts have been made to extend this methodology toward more refined models of systems and behaviors, going from *discrete* to *timed* and then to *hybrid* (discrete-continuous) systems. Such extensions involve departures from each of the above features.

The first departure consists in replacing the discrete time domain by the dense and metric domain $\mathbb{R}$. Behaviors of this type are generated by timed system models such as timed automata [1] and similar formalisms that can express the *durations* of discrete processes. Natural extensions of LTL to handle dense time are logics such as MTL [14] or MITL [2] which can express requirements concerning the time elapsing between events. Timed behaviors can be viewed as either *Boolean signals*, which are functions from the real time axis to $\mathbb{B}^n$, or as *timed words* consisting of instantaneous events, taken from some alphabet, separated by real time durations, see [4]. The second departure, motivated by the application of verification and monitoring techniques to continuous and hybrid systems, consists of letting predicates over the reals, such as

[1] In a probabilistic setting one assigns probabilities to the satisfaction of a formula but this is done with respect to a *set* of behaviors, while the satisfaction by *individual* behaviors remains Boolean.

inequalities, play the role of atomic propositions[2] and thus specify properties of real-valued signals. In [15] the logic STL (*signal temporal logic*) which combines the dense time modalities of MITL with such numerical predicates has been introduced, along with a monitoring tool [17, 19] for deciding satisfaction of such properties by time-stamped traces produced by numerical simulators.

However, the third premise of a yes/no answer is not fully compatible with *quantitative* entities in the continuous domain where real-valued *distance* functions play an important role. As an illustration, consider the inequality $x \leq c$. Boolean satisfaction does not make a difference between $x = c+\epsilon$ and $x >> c$ as both cases are classified as violating the property. Likewise, one cannot distinguish between marginal satisfaction by $x = c - \epsilon$ and a more robust satisfaction by $x << c$. The same criticism applies to satisfaction of timing constraints: a requirement that some event occurs within t time can be violated by its occurrence at $t + \epsilon$, at $t' >> t$ as well as by its complete absence, and a yes/no answer cannot tell the difference.

These issues are extremely important in the continuous setting because such systems are subject to noise and numerical errors, not to mention the inherent approximative character of mathematical models of natural phenomena. Consequently, parameters appearing in system descriptions such as differential equations, as well as in formulae, are often a result of guessing and estimation and should *not* be regarded as representing universal constants. Thus if a property is violated in a marginal way as in $x = c + \epsilon$, satisfaction can still be achieved by slight modifications in the property, in the behavior or in the parameters of the generating system. In fact, the use of temporal logic in a *scientific* context, as in *systems biology* is methodologically different from its use in the *engineering* context of system design. In biology, it is often the case that the role of the temporal formula is to provide a *succinct abstract model* of the *observed* behavior of a complex network of chemical reactions [10, 3]. The question there is not whether the system model satisfies a *given* specification but rather to find a formula, as semantically-tight as possible, compatible with the system model or with a set of observed behaviors. In such a model-search process it is important to know how close we are to a satisfactory model and which parameters should be changed (and in which direction) in order to approach it.

These observations have led several researchers to look more closely at the notion of *robust satisfiability* by continuous signals of properties expressed in STL or similar formalisms [11, 22, 23].[3] Fainekos and Pappas [11] define the notion of *robustness degree* as a real number associated with a property-behavior pair, based on, roughly speaking, the *distance* between the behavior and the (boundary of) the set of all behaviors that satisfy the property. This measure is more positive when the behavior is deeper inside the set of satisfying behaviors and more negative the further is the behavior outside that set. Hence it satisfies a natural notion of *soundness* of such a robustness measure, namely,

[2] Such predicates are used, of course, also in more conventional applications of temporal logic to programs with *numerical variables* and also in similar logics introduced and used in biology [3, 10]. All such logics are quantifier-free fragments of first-order temporal logic.

[3] The study of robust satisfaction and, more generally, of *multi-valued* and *quantitative* interpretation of logical formalisms is, of course, very old. We focus on works relevant to our motivation, *sensitive monitoring* of *continuous signals.*

being positive for signals that satisfy the property and negative for violating signals. They propose an inductive procedure for computing the robustness degree, as well as a recent tool [12]. Their approach is behavior oriented, indicating how much should the *signal* be modified in order to satisfy or violate the property.

The measures introduced by Rizk et al. [22, 23] are more property oriented and intend to assess how far a given formula, written in some variant of LTL[4] augmented with numerical predicates, is from being an adequate description of a given simulation trace. Their real-valued degree of satisfaction is obtained first by replacing all constants in the formula by parameters and hence viewing all formulae admitting the same form as points in an Euclidean parameter space. They compute the set of all points that correspond to formulae satisfied by a given trace (or set of traces) and define a real-valued satisfaction/violation degree based on the distance between the original formula and the set of satisfied formulae. They use this measure to guide a search in the parameter space. Their approach has been integrated into the BIOCHAM tool [6] and has been applied to numerous biological examples.

In this paper we extend these works in several directions. First we propose an alternative quantitative semantics for MITL/STL which focuses on the robustness of satisfaction with respect to *time*. We then show how these two robustness measures can be combined into a generalized robustness measure which captures both space and time, from which both the space robustness of [11] and our time robustness are obtained as special cases. Then we present an efficient dense-time algorithm for computing these measures for piecewise-linear signals. Finally we extend the algorithm to compute the sensitivity of these measures with respect to parameter variations thus paving the way for the extension of sensitivity-based parameter-space exploration methodology of [7] to handle STL formulae.

2 Logics for Real-Valued Signals

In this section, we partly and briefly recall the framework set in [15, 17] to define an appropriate logic for real-valued continuous time signals. In the rest of the paper, we implicitly assume the existence of a system under study whose state is described by a set of n variables $V = \{x_1,\ x_2, \ldots, x_n\}$. The domain of valuation of V is denoted by $\mathbb{D} = \mathbb{D}_1 \times \mathbb{D}_2 \times \ldots \times \mathbb{D}_n$. The domain $\mathbb{R}$ is the set of real numbers, $\mathbb{B} = \{\text{true},\ \text{false}\}$ is the Boolean domain and the time set is $\mathbb{T} = \mathbb{R}_{\geq 0}$. As a preparation for the real-valued semantics we will view $\mathbb{B}$ as $\{-1, +1\}$ rather than $\{0, 1\}$. Disjunction and conjunction are realized as $\max$ and $\min$ and negation as minus. This way the passage to the quantitative semantics of [11] will be immediate. A trace (or signal or behavior) w describing an evolution of the system is a function from $\mathbb{T}$ to $\mathbb{D}$. We define a set $P = \{p_1, p_2, \ldots, p_n\}$ of projectors, so that for a given trace w, $p_i(w[t]) = x_i[t]$ for all t. If the context is not ambiguous, we write $p_i[t]$ instead of $p_i(w[t])$.

[4] Their treatment of time, interpreting the *next* operator as referring to the next integration step, is too implementation-dependent to serve as a solid basis for defining time robustness.

2.1 Metric Interval Temporal Logic (MITL)

We first consider continuous-time Boolean signals, i.e., when $\mathbb{D} = \mathbb{B}^n$. A popular logic to characterize such timed behaviors is the *Metric Interval Temporal Logic* (MITL) [2] whose grammar is given by:

$$\varphi := p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \,\mathcal{U}_{\mathcal{I}}\, \varphi_2 \tag{1}$$

where φ, φ_1 and φ_2 are MITL formulae, $p \in P = \{p_1, p_2, \ldots, p_n\}$ and $\mathcal{I}$ is an interval of the form $\mathcal{I} = (i_1, i_2)$, $(i_1, i_2]$, $[i_1, i_2)$ or $[i_1, i_2]$, where $i_1 < i_2$ are in $\mathbb{T}$. For t in $\mathbb{T}$, $t + \mathcal{I}$ is the set $\{t + t' \mid t' \in \mathcal{I}\}$. Traditionally the satisfaction of an MITL formula φ by a trace w at time t is denoted by $(w, t) \vDash \varphi$. We will use instead the characteristic function $\chi(\varphi, w, t)$ which is 1 when $(w, t) \vDash \varphi$ and -1 otherwise.

Definition 1 (Semantics). *The characteristic function of an MITL formula relative to a trace w at time t is defined inductively as*

$$\chi(p, w, t) = p[t] \tag{2}$$
$$\chi(\neg\varphi, w, t) = -\chi(\varphi, w, t) \tag{3}$$
$$\chi(\varphi_1 \wedge \varphi_2, w, t) = \min(\chi(\varphi_1, w, t), \chi(\varphi_2, w, t)) \tag{4}$$
$$\chi(\varphi_1 \,\mathcal{U}_{\mathcal{I}}\, \varphi_2, w, t) = \max_{t' \in t + \mathcal{I}} \min(\chi(\varphi_2, w, t'), \min_{t'' \in [t, t']} \chi(\varphi_1, w, t'')) \tag{5}$$

Note again that the semantics of *until* is equivalent to the more familiar notation:

$$(w, t) \vDash \varphi_1 \,\mathcal{U}_{\mathcal{I}}\, \varphi_2 \Leftrightarrow \exists t' \in t + \mathcal{I} \text{ s.t. } (w, t') \vDash \varphi_2 \text{ and } \forall t'' \in [t, t'], (w, t'') \vDash \varphi_1$$

where dense Boolean operations are replaced by dense $\min$ and $\max$. From the basic operators of MITL, other standard operators can be defined such as *eventually* and *always*: $\Diamond_{\mathcal{I}}\varphi \triangleq \text{true}\ \mathcal{U}_{\mathcal{I}}\ \varphi$, $\Box_{\mathcal{I}}\varphi \triangleq \neg\Diamond_{\mathcal{I}}\neg\varphi$.

2.2 Signal Temporal Logic (STL)

Signal temporal logic (STL) [15] allows one to apply MITL-like reasoning to traces over $\mathbb{D} = \mathbb{R}^n$. The connection is done via a finite set $M = \{\mu_1, \ldots, \mu_k\}$ of predicates (Booleanizers), each mapping $\mathbb{R}^n$ to $\mathbb{B}$. For a given $1 \leq j \leq k$, the predicate μ_j is of the form $\mu_j \equiv f_j(x_1, x_2, \ldots, x_n) \geq 0$ where f_j is some real-valued function.

Definition 2. *We call x_i the* primary *signals of w and call their images by f_j* secondary *signals that we denote by $\{y_1, \ldots, y_k\}$.*

The syntax is thus identical to MITL except for the predicates in M replacing the atomic propositions in P. The semantics is identical to that of Definition 1 except for the base case (we interpret $\text{sign}(0)$ as 1):

$$\chi(\mu, w, t) = \text{sign}(f(x_1[t], \ldots, x_n[t])) \tag{6}$$

Figure 1 illustrates the semantics of the STL formulae $\Diamond_{[1,2]}(x_1 + 2x_2 - 2 \geq 0)$ relative to a two-dimensional real-valued signal. The methodology developed in [15, 17] for

deciding satisfaction of an STL formula φ by a real-valued signal is based on assigning to each sub-formula ψ of φ a *satisfaction signal*, that is a Boolean signal whose value at t is equal to $\chi(\psi, w, t)$. The computation goes bottom up (and backwards, for the future fragment of the logic) starting from the primary signals from which the secondary signals and their Booleanizations are computed, and then climbing up the parse tree of φ until the satisfaction signal of the top formula is computed. This procedure is similar in spirit to the novel translation from MITL to timed automata [16] based on the compositional principles of *temporal testers* initiated in [13, 21] where each formula is associated with a transducer which computes the satisfaction signal of the formula based on those of its sub-formulae. In Section 4 we present a monitoring procedure for the quantitative semantics. With the flexibility of the definition of STL we can express

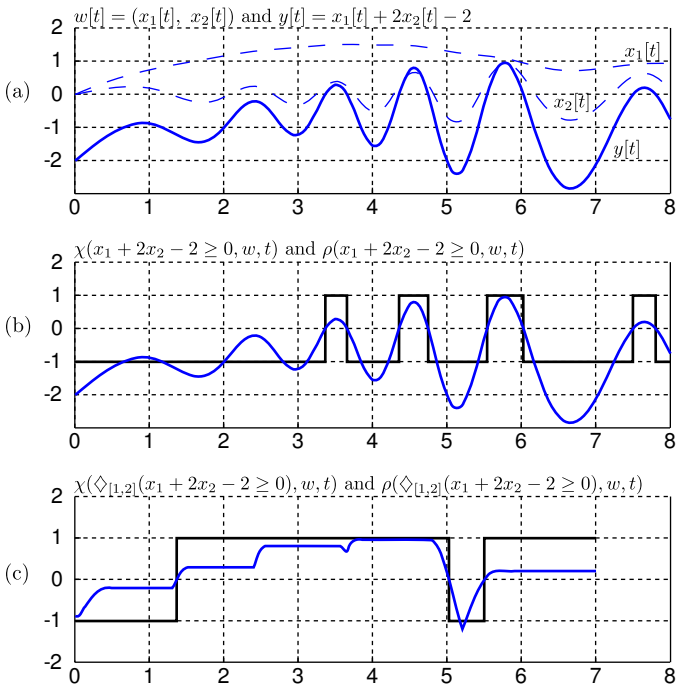

Fig. 1. A two-dimensional real-valued signal w and the time evolution of $\chi(\varphi, w, t)$ and $\rho(\varphi, w, t)$ for $\varphi \triangleq \Diamond_{[1,2]}(x_1 + 2x_2 - 2 \geq 0)$. (a) the primary signals x_1, x_2 and the secondary signal $y = x_1 + 2x_2 - 2$; (b) the truth value and spatial robustness of the sub-formula corresponding to $y \geq 0$; (c) the truth value and spatial robustness of $\Diamond_{[1,2]} y \geq 0$ (notice the smoothing effect of the non-punctual eventually operator).

a rich collection of relevant temporal properties for continuous signals. However, it has two drawbacks: first, the values of the primary and secondary signals at *different* time instances cannot "communicate" before being Booleanized and then handled by the temporal operators. This limitation can be alleviated by letting predicates use also the

shift operator (which is a punctual version of $\Diamond$). Although our implementation allows this feature we will assume here only simple point-wise predicates in order to focus on the second drawback mentioned in the introduction: the loss of quantitative information due to Booleanization. In the next section, we propose several quantitative semantics for STL reflecting the *robustness*, in space and time, of satisfaction or violation.

3 Quantitative Semantics

The first quantitative measure of satisfaction that we present is a simple reformulation of the spatial robustness degree of [11], after which we proceed to a novel measure of *temporal* robustness and finally to a generalized measure that combines both. The definitions of all these measures are identical to Definition 1 except for the base cases.

3.1 Space Robustness

Definition 3 (Space Robustness). *The space robustness of an STL formula relative to a trace w at time t, denoted by $\rho(\varphi, w, t)$, is defined inductively as*

$$\begin{aligned}
\rho(\mu, w, t) &= f(x_1[t], \ldots, x_n[t]) \text{ where } \mu \equiv f(x_1, \ldots, x_n) \geq 0 \\
\rho(\neg\varphi, w, t) &= -\rho(\varphi, w, t) \\
\rho(\varphi_1 \wedge \varphi_2, w, t) &= \min(\rho(\varphi_1, w, t), \rho(\varphi_2, w, t)) \\
\rho(\varphi_1 \,\mathcal{U}_{\mathcal{I}}\, \varphi_2, w, t) &= \max_{t' \in t+\mathcal{I}} \big(\min(\rho(\varphi_2, w, t'), \min_{t'' \in [t,t')} \rho(\varphi_1, w, t'')\big)
\end{aligned}$$

It is not hard to see that the definition above is sound in the sense of [11]: $\chi(\varphi, w, t) = \text{sign}(\rho(\varphi, w, t))$. Figure 1 illustrates the behavior of this measure. Note that taking the $\min$ or $\max$ of a signal over a temporal window is an alternative way (compared to averaging) of "smoothing" it. In Section 4 we will present an efficient algorithm for computing this semantics on piecewise-linear signals, the natural interpretation of the output of numerical simulators. This measure captures the robustness of the satisfaction to noise in w. Let the point-wise distance between two finite signals of the same length be $||y - y'|| = \max_t |y[t] - y'[t]|$. The following is a reformulation of the results of [11]:

Theorem 1 (Property of Space Robustness). *If $\rho(\varphi, w, t) = r$ then for every w' in which every secondary signal satisfies $||y_j - y'_j|| < r$, $\chi(\varphi, w, t) = \chi(\varphi, w', t)$.*

3.2 Time Robustness

While this semantics captures the robustness of satisfaction with respect to *point-wise* changes in the value of the signal or in constants appearing in the predicates, it does not fully capture the effect of changes in the constants appearing in the *temporal operators* of the formula nor in changes in the signal along the time axis. Such changes are often expressed using a *retiming function*, a monotone function $\alpha : \mathbb{T} \to \mathbb{T}$ which transforms a signal x to x' by letting $x'[t] = x[\alpha(t)]$. Figure 3.2 shows three different signals

satisfying $\rho(\varphi, w_1, 0) = \rho(\varphi, w_2, 0) = \rho(\varphi, w_3, 0) > 0$ for the formula $\varphi \triangleq \Diamond_{[a,b]}(x > 0)$. Intuition tells us, however, that w_1 satisfies φ more robustly being positive during a large part of the $[a, b]$ interval while w_2 has only a short positive spike and w_3 almost misses the deadline for satisfying the positivity condition. All those signals have the same value of ρ because they admit the same maximal value in the interval and the point-wise space robustness cannot account for these differences. For the same reason, it cannot capture the similarity between w_3 and the property-violating signal w_4, obtained from w_3 by a slight shifting in time, that is, $w_4[t] = w_3[t - \varepsilon]$.

This observation motivates our definition of *time robustness* which indicates the effect on satisfaction of *shifting* events in *time*, where the term *event* refers to rising and falling edges in Boolean signals, the moments where certain secondary signals cross a threshold and their predicates change their truth value. Since the notion of a change in a truth value is discrete, we define time robustness with respect to MITL and Boolean signals but the definition applies as well to the standard Boolean semantics of STL where Booleanizers replace atomic propositions.

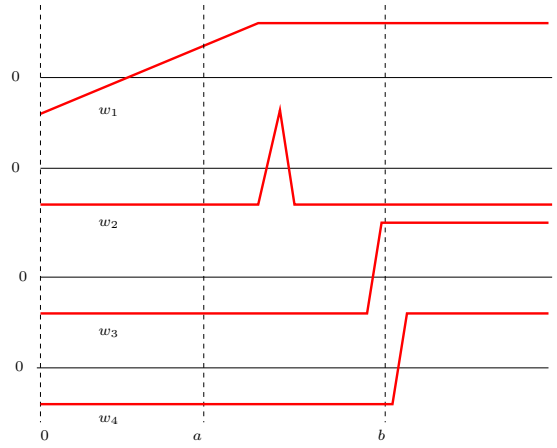

Fig. 2. Limitations of the point-wise quantitative semantics: signals w_1, w_2 and w_3 are considered as satisfying $\Diamond_{[a,b]}(x > 0)$ from $t = 0$ at the same degree, while the similarity between w_3 and the violating w_4 is not captured.

Definition 4 (Time Robustness). *The left and right time robustness of an MITL formula φ with respect to a trace w at time t are defined inductively by letting*

$$\theta^-(p, w, t) = \chi(\varphi, w, t) \cdot \max\{d \geq 0 \text{ s.t. } \forall t' \in [t - d, t],\ \chi(\varphi, w, t') = \chi(\varphi, w, t)\}$$
$$\theta^+(p, w, t) = \chi(\varphi, w, t) \cdot \max\{d \geq 0 \text{ s.t. } \forall t' \in [t, t + d],\ \chi(\varphi, w, t') = \chi(\varphi, w, t)\}$$

and then applying to each of (θ^-, θ^+) the rules (3-4) as in Definition 1.

Figure 3 illustrates the time robustness of the satisfaction of p by a Boolean signal. Note that there are (unavoidable) discontinuities in the evolution of these measures at

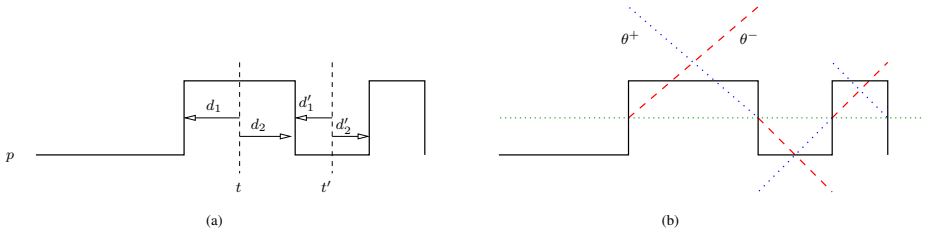

Fig. 3. (a) Illustration of time robustness of a propositional formula p with respect to a Boolean signal: $\theta^-(p,w,t) = d_1$; $\theta^+(p,w,t) = d_2$; $\theta^-(p,w,t') = -d'_1$; $\theta^+(p,w,t) = -d'_2$; (b) The evolution of θ^- and θ^+ with time.

rising/falling edges and that $\theta^-(p,w,t) + \theta^+(p,w,t)$ is constant inside an interval in which the truth value of p is uniform. The following property holds naturally for atomic propositions and is preserved by temporal operators:

Theorem 2 (Property of Time Robustness). *If $\theta^-(\varphi,w,t) = s$ (resp. $\theta^+(\varphi,w,t) = s$) then for any signal w' obtained from w by shifting events to the right (resp. to the left) by less than s, we have $\chi(\varphi,w,t) = \chi(\varphi,w',t)$.*

3.3 Combined Space-Time Robustness

We can now move to a combined space-time robustness which reflects trade-offs between space and time robustness: the same signal will be more robust temporally if we are ready to compromise its spatial robustness and vice versa. The space-time robustness of an STL formula φ with respect to a trace w at time t is a family of function pairs $\{\theta_c^+, \theta_c^-\}_{c\in\mathbb{R}}$ which map every spatial robustness c to left/right temporal robustness. For an STL predicate μ, let $\chi_c(\mu,w,t) = \text{sign}(\rho(\mu,w,t) - c)$, that is, a Boolean whose value is positive if and only if the robustness of μ for w at t is at least c.

Definition 5 (Space-Time Robustness). *The temporal robustness of a formula φ relative to a spatial robustness c is obtained by letting*

$$\theta_c^-(\mu,w,t) = \theta^-(\chi_c(\mu,w,t))$$
$$\theta_c^+(\mu,w,t) = \theta^+(\chi_c(\mu,w,t))$$

and then applying the rules of Definition 4.

Geometrically, these functions define the basis of the largest rectangle of height c that can be constructed around t while remaining below the secondary signal associated with μ, as illustrated in Figure 4. In fact, the set of realizable triples (ρ,θ^-,θ^+) represents the possible trade-offs (a kind of Pareto curve) between these measures. It could equivalently be captured by a function ρ_{d^-,d^+} which maps time robustness to space robustness. The previously-defined space and time robustness measures are obtained as the special cases: $\rho = \rho_{0,0}$, $\theta^- = \theta_0^-$ and $\theta^+ = \theta_0^+$.

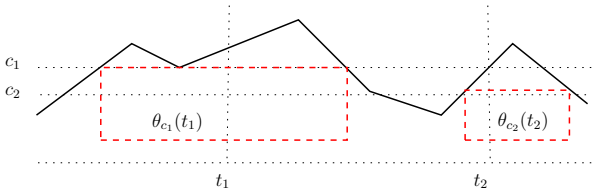

Fig. 4. Illustration of the combined time-space robustness.

4 Computing Robustness Degrees

In this section we extend the algorithm of [15] for deciding the satisfaction of an STL formula by a given signal w, to compute the quantitative semantics. We focus on space robustness ρ but the computation for θ or θ_c (for any value of c) is similar. The robustness function $\rho(\varphi, w, \cdot)$ is computed inductively on the structure of the formula, beginning with the computation of the secondary signals based on the primary signals in w and the predicates in M. Thus we need to compute the right-hand side terms of the semantics in Definition 1 which reduces to the following subproblems corresponding, respectively, to operators $\neg$, $\wedge$ and $\mathcal{U}_\mathcal{I}$:

1. Given $y : \mathbb{T} \to \mathbb{R}$, compute $z : \mathbb{T} \to \mathbb{R}$ such that $\forall t \in \mathbb{T}$, $z[t] = -y[t]$;
2. Given $y, y' : \mathbb{T} \to \mathbb{R}$, compute $z : \mathbb{T} \to \mathbb{R}$ such that
$$\forall t \in \mathbb{T},\ z[t] = \min(y[t], y'[t]) \tag{7}$$
3. Given $y, y' : \mathbb{T} \to \mathbb{R}$ and an interval $\mathcal{I}$, compute $z : \mathbb{T} \to \mathbb{R}$ such that
$$\forall\, t \in \mathbb{T},\ z[t] = \max_{\tau \in t+\mathcal{I}} (\min(y'[\tau],\ \min_{s \in [t,\tau]} y[s])) \tag{8}$$

The first of these being trivial, we focus on the second and third. The difficulty lies in the fact that we compute *functions* and not only single values at specific time instants. Dealing with continuous time and space, a natural input for our algorithm is a sequence of time-stamped values of w obtained via variable step-size numerical simulation that we interpret as a *piecewise-linear* function by linear interpolation. More precisely, we assume that a secondary signal y is a piecewise-affine function of time with a finite sequence of points $\{t_k\}_{1<k<n_y}$ where its derivative changes. We do not require continuity at these points (which allows us to deal with hybrid behaviors) but we assume that y is right-continuous and admits a right-derivative noted $dy[t] \triangleq \lim_{\varepsilon \to 0} \frac{y[t+\varepsilon]-y[t]}{\varepsilon}$, which is just its slope at time t. As usual, if y is not continuous at t, we note $y[t^-]$ (resp. $y[t^+]$) its limit to the left (resp. to the right) of t. Finally, we extend the trace to be unbounded by a constant extrapolation before t_0 and after t_{n_y}, that is, $y[t] = y[t_0]$ for $t < t_1$ and $y[t] = y[t_{n_y}]$ for $t > t_{n_y}$.[5]

It is not hard to see that if y and y' are piecewise-affine and represented by a finite number of sampling point so is any z satisfying (7) or (8). Hence it is sufficient to know

[5] There are various other ways to interpret temporal logic over finite traces, such as the weak semantics of [9] and other solutions surveyed in [17]. This topic is orthogonal to the rest of the paper.

its values and its slopes at a finite sequence of time instances $\{r_k\}_{1\leq k\leq n_z}$ such that z is continuous and dz is constant on each interval $[r_k, r_{k+1})$. We note

$$\text{NextEvent}(z,t) = \begin{cases} \infty \text{ if } (z,dz) \text{ is continuous for } r > t \\ \min\{r > t \mid (z[r^-], dz[r^-]) \neq (z[r^+], dz[r^+])\} \text{ otherwise} \end{cases}$$

thus z can be computed incrementally by the generic Algorithm 1. An interesting feature of this approach is that it can easily be adapted to work *online* where input w is revealed progressively. Indeed, each time a new pair $(z(r_k), dz(r_k))$ is computed, the algorithm can be paused to wait for additional input data w needed to compute r_{k+1} and $(z(r_{k+1}), dz(r_{k+1}))$. To implement Algorithm 1, we need to implement the intializa-

Algorithm 1 An iterative algorithm to compute the robustness z of the conjunction or the *until* of two secondary signals y and y'

1: **Init** $r_1, k = 1$
2: **Repeat**
3: Compute $(z[r_k], dz[r_k])$ from (y, y')
4: Compute $r_{k+1} = \text{NextEvent}(z, r_k)$ from (y, y')
5: Let $k = k + 1$
6: **Until** $r_k = \infty$

tion (line 1), the computation of (z, dz) at a given time instant (line 3) and the NextEvent function (line 4) based on the representation of y and y' sampled at $\{t_k\}_{1\leq k\leq n_y}$ and $\{t'_k\}_{1\leq k\leq n_y}$.

4.1 Conjunction: $z[t] = min(y[t], y[t'])$

Initialization and computation of (z, dz). In the case of the conjunction, we initialize r_1 to $\min(t_1, t'_1)$ and we note that computation of (z, dz) satisfying (7) is trivial except when $y[t] = y'[t]$. In that case it can be seen easily though that $dz[r] = \min(dy[z], dy'[r])$ (the $\min$ operator preserves the right-derivative) so that if we extend the $\min$ function with the lexicographic order, we have

$$\forall t \in \mathbb{T},\ (z[t], dz[t]) = \min\{\ (y[t], dy[t]),\ \ (y'[t], dy'[t])\ \} \tag{9}$$

Computation of next event. Observe first that since we know y and y', we know their NextEvent functions. Then, the slope of z can be discontinuous in $r > r_k$ only if r is a time event for y or y' or if $y[r] = y'[r]$. Thus there are three possibilities:

1. $(z[r_k], dz[r_k]) = (y[r_k], dy[r_k])$ then
$\text{NextEvent}(z, r_k) = \min\{\ \text{NextEvent}(y, r_k),\ \ \arg\min_{t>r_k}\{y[t] = y'[t]\}\}$
2. $(z[r_k], dz[r_k]) = (y'[r_k], dy'[r_k])$ then
$\text{NextEvent}(z, r_k) = \min\{\ \text{NextEvent}(y', r_k),\ \ \arg\min_{t>r_k}\{y[t] = y'[t]\}\}$
3. $(z[r_k], dz[r_k]) = (y[r_k], dy[r_k]) = (y'[r_k], dy'[r_k])$ then
$\text{NextEvent}(z, r_k) = \min\{\ \text{NextEvent}(y, r_k),\ \text{NextEvent}(y', r_k)\}$

The resulting computation of the conjunction is illustrated on Figure 5.

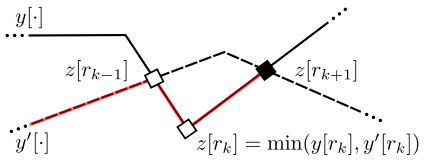

Fig. 5. Computation of $\min(y, y')$.

4.2 The *Until* Operator: $z[t] = \max_{\tau \in t+\mathcal{I}}(\min(y'[\tau], \min_{s\in[t,\tau]} y[s]))$

The computation for the *until* operator involves the detection of the change in the maximal or minimal value of a signal over a moving time window. We note $\underline{y}[t,\tau] = \min_{s\in[t,\tau]} y[s]$, so that we have to compute

$$\forall\, t \in \mathbb{T},\ z[t] = \max_{\tau\in t+\mathcal{I}}(\min(y'[\tau], \underline{y}[t,\tau])) \tag{10}$$

The value $z[t]$ is provided either by y or by y' at some time instant in $[t, t+i_2]$. The following result, that we use to compute (z, dz) and the NextEvent function, shows that if $z[t]$ is not provided by y', then it has to be $\underline{y}[t, t+i_1]$.

Lemma 1. *We say that τ is an admissible time for y' iff $y'[\tau] \leq \underline{y}[t,\tau]$. If there is no admissible time for y', then $z[t] = \underline{y}[t, t+i_1] = \min_{s\in[t,t+i_1]} y[s]$.*

Proof. Since there is no admissible time for y', there must be $t^* \in [t, t+i_2]$ such that $z[t] = y[t^*]$. Moreover, $y[t^*]$ has to be equal to $\min_{s\in[t,\tau]} y[s]$ for some τ in $[t+i_1, t+i_2]$. Since $[t, t+i_1] \subseteq [t,\tau]$, $z[t] = y[t^*] \leq \min_{s\in[t,t+i_1]} y[s]$. Furthermore, $z[t] = y[t^*] = \max_{\tau\in t+\mathcal{I}}(\min_{s\in[t,\tau]} y[s])$ and since $t+i_1 \in t+\mathcal{I}$, $z[t] \geq \min_{s\in[t,t+i_1]} y[s]$ so that $z[t] = \min_{s\in[t,t+i_1]} y[s]$.

Initialization and computation of (z, dz). Since $(z[t], dz[t])$ depends on the values of y and y' on the interval $[t, t+i_2]$, the first time event for z is $r_1 = \min(t_1 - i_2, t'_1 - i_2)$. Then to compute $z[t]$ for a given t, we first scan chronologically the values of y on $[t, t+i_1]$ to compute their minimum $\underline{y}[t, t+i_1]$. Then we scan the values of y' and y for τ in $[t+i_1, t+i_2]$, updating $\underline{y}[t,\tau]$, then getting the minimum with $y'[\tau]$ and finally the maximum with the value obtained before τ. At the end, we get the value $z[t]$.

In the process, we also compute $dz[t]$ starting by initializing it to $dy[t]$. Then, each time a new value for the minimum of y is found for some τ in the interval $(t, t+i_1)$, we compare the current value for $dz[t]$ with 0 (because τ does not depend on t) and keep the minimum. For $\tau = t+i_1$, we compare the current value for $(z[t], dz[t])$ with $(y'[t+i_1], dy'[t+i_1])$ and $(y[t+i_1], dy[t+i_1])$ and keep the minimum (in the lexicographic sense). For τ in the interval $(t+i_1, t+i_2)$, as long as we do not find an admissible time for y', we only update $dz[t]$ if $y[\tau]$ is equal to the current value of $z[t]$, in which case we set to $dz[t] = min(0, dz[t])$. When a new admissible time τ is found,

$(z[t], dz[t])$ is updated to $\max((y'[\tau], 0), (z[t], dz[t]))$. Finally, if $t+i_2$ is admissible, we let $(z[t], dz[t]) = \max((y'[\tau], dy'[\tau]), (z[t], dz[t]))$. We note $\arg z[t]$ the time instant in $[t, t+i_1]$ when the value of $(z[t], dz[t])$ is determined.

Computation of next event Assuming that we computed $(z[r_k], dz[r_k])$, we need to compute the minimum time $r_{k+1} > r_k$ such that z is discontinuous in r_{k+1} and/or $dz[r_{k+1}]$ is different from $dz[r_k]$. Firstly, an event can occur at $r > r_k$ due to an event for y or y' at r_k, $r_k + i_1$ or $r_k + i_2$, i.e. if

$$\begin{aligned} r = \min\{&\text{NextEvent}(y, t), \text{NextEvent}(y', t') \text{ such that} \\ &t \in \{r_k,\ r_k + i_1, r_k + i_2\}, t' \in \{r_k + i_1,\ r_k + i_2\}\} \end{aligned} \tag{11}$$

The second possibility of event depends on whether the value of $(z[r_k], dz[r_k])$ comes from y or y'. If there is no admissible time for y', we showed that $z[r_k]$ is $\underline{y}[r_k, r_k + i_1]$ so an event can occur when $\underline{y}[r, r + i_1]$ and/or its derivative is discontinuous. We can show that this can happen only if r satisfies:

$$\begin{aligned} &\min\{y[t_j], y[t_j^-] \mid r < t_j < r + i_1\} = \min(y[r], y[r + i_1]) \\ \text{or } &\min\{y[t_j], y[t_j^-] \mid r < t_j < r + i_1\} > y[r] = y[r + i_1] \end{aligned} \tag{12}$$

Now, whether there are admissible times for y' or not, we have to monitor the appearance of new admissible (adm.) times. This can happen only if r satisfies:

$$\begin{aligned} &y[r] = \min\{y'[t'_j], y'[t'^-_j] \mid r + i_1 < t'_j < r + i_2 \text{ with } t'_j \text{ not adm. for } r_k\} \\ \text{or } &y'[r + i_1] = y[r] \text{ with } r + i_1 \text{ not adm. for } r_k \\ \text{or } &y'[r + i_2] = \underline{y}[r, r + i_2] \end{aligned} \tag{13}$$

Finally, we have to monitor discontinuities in the maximum values of y' among existing admissible times. They can happen only if

$$\begin{aligned} &r+i_1 \text{ is adm. and } y'[r] = \max\{y'[t'_j], y'[t_j^-] \mid r+i_1 < t'_j(\text{adm.}) < r+i_2\} \\ \text{or } &r+i_2 \text{ is adm. and } y'[r+i_2] = \max\{y'[t'_j], y'[t'^-_j] \mid r+i_1 < t_j(\text{adm.}) < r+i_2\} \\ \text{or } &y'[r+i_1] = y'[r+i_2] \text{ with } r+i_1,\ r+i_2 \text{ adm.} \end{aligned} \tag{14}$$

Then the expressions (11-14) provides conservative conditions for $r > r_k$ to be NextEvent(z, r_k). The algorithm compute the minimum $\tilde{r}_k > r_k$ satisfying one of these conditions, then $(z[\tilde{r}_k], dz[\tilde{r}_k])$ and iterates with $r_k = \tilde{r}_k$ until (z, dz) is found to be discontinuous in $\tilde{r}_k$. Figure 6 illustrates the process.

4.3 Computational Cost

The complexity of the robustness computation for a given signal and a formula is clearly linear with respect to the computational cost of Algorithm 1, the constant being the size of the formula. The cost (time and space) of algorithm 1 depends on the number of events generated. In the case of conjunction, there can be as many as $n_y + n_{y'} + \max(n_y, n'_y)$ (the third term counting the maximal number of intersections between y and y') so it is linear in the number of samples in y and y'.

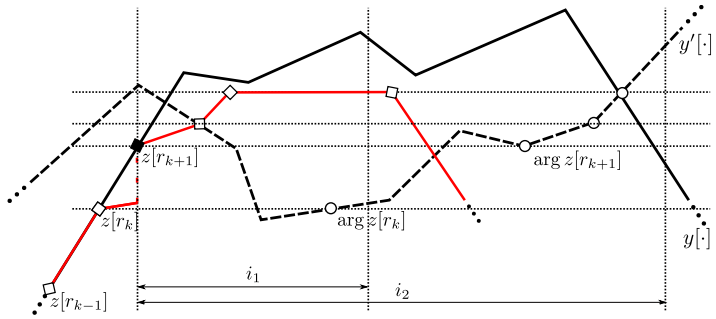

Fig. 6. Computation of *until* robustness. Here, $\arg z$ represents the point that determines z. For $\tau < r_k$, there is no admissible time for y' and $z[\tau] = y[\tau]$. The following admissible time is $r_k + i_1$, since $y[\tau] = y'[\tau + i_1]$ (second condition of (13)). At r_{k+1}, another time becomes admissible, which causes z to be discontinuous (first condition of (13)).

In the case of the *until* operator, this number is more difficult to estimate due to the complex inter-dependance of values of y and y' in the intervals $[t, t + i_2]$. We conjecture that this bound is also linear in $n_y + n_{y'}$. Then each time step requires the computation of (z, dz) for a given interval $[t, t + i_2]$ involving scanning and comparing the values of y and y' in this interval. The worst complexity for this operation is of the order of the sum of the maximum number of sampling points of y and y' that $[t, t+i_2]$ can contain. We write this number $n_{(y \cup y') \cap \mathcal{I}}$, so that our conjectured complexity is $\mathcal{O}\big((n_y + n_{y'}) \times n_{(y \cup y') \cap \mathcal{I}}\big)$. In practice, we found that the complexity is usually better since the value of z may have much less changes due to the fact that the maximum/minimum of a signal over a moving time window is likely to remain constant for long periods.

We compared our implementation with TaLiRo [12], the only other known tool for computing robustness degrees of temporal formulae. Their algorithm, for which the implementation details are not provided, appeared experimentally to behave linearly in the size of the input signals but, at least in some cases, exponentially (in time and memory) in the size of the formula. For the same data and formulae, our algorithm was behaving as (or better than) the above analysis suggests.[6]

5 Robustness Sensitivity

In [8, 7] we have developed a methodology, based on numerical simulation and sensitivity analysis, to explore the parameter space of a dynamical system in order to determine the region in this space which induces some qualitative behavior. This work has been restricted so far to simple reachability properties and the development in this paper extends its applicability to the whole range of temporal properties, with sensitivity defined as follows.

[6] See `http://www-verimag.imag.fr/~donze/breach_page.html` for experimental data.

Assume that the robust satisfaction of a formula φ by a signal w is parameterized by some $\lambda \in \mathbb{R}$, i.e., for $t \in \mathbb{R}$, its value is $\rho(\varphi, w, t, \lambda)$, and that it is differentiable with respect to λ, with derivative noted $d_\lambda \rho(\varphi, w, t, \lambda)$. As a simple illustration, consider the predicate μ: $x_1 + 2x_2 - \lambda \geq 0$ with the corresponding secondary signal y. We have $\rho(x_1 + 2x_2 - \lambda \geq 0, w, t, \lambda) = y[t, \lambda] = x_1[t] + 2x_2[t] - \lambda$, which is differentiable with respect to λ, with $d_\lambda y[t, \lambda] = -1$. Another common situation is when the signal results from the simulation of a system with an uncertain parameter λ. Then w is a function of λ and we can get the derivative of the primary signals from a sensitivity-aware simulator and deduce those of the secondary signals by applying the chain rule.

Having defined the robustness sensitivity to λ for some base formulae (supposedly for all λ and t), we remark that our algorithm presented above can be easily adapted to compute the robustness sensitivity of any formula built from these base formulae. As for the robustness, the core of the algorithm needs only to implement the derivative of some function z satisfying $z[t, \lambda] = -y[t, \lambda]$, the relation (7) or the relation (8) including λ, corresponding to $\neg$, $\wedge$ and $\mathcal{U}_\mathcal{I}$. We first observe that if y and y' are differentiable with respect to λ, then $z = \min(y, y')$ is differentiable everywhere except when $y[t, \lambda] = y'[t, \lambda]$ for some $t \in \mathbb{T}$. However, it has a right- and left-derivative at this point which are

$$d^+_\lambda z[t, \lambda] = \min(d^+_\lambda y[t, \lambda], d^+_\lambda y'[t, \lambda]) \text{ and } d^-_\lambda z[t, \lambda] = \max(d^-_\lambda y[t, \lambda], d^-_\lambda y'[t, \lambda]) \tag{15}$$

Thus we can adapt Algorithm 1 to compute the right- and left-derivatives of z in a way similar to the way we compute dz. In addition to computing $(z[r_k]), dz[r_k])$, we compute $(d^+_\lambda z[r_k, \lambda], d^-_\lambda z[r_k, \lambda])$: each time z is updated with the comparison of two signals taking the same value, we update $d^+_\lambda z$ and $d^-_\lambda z$ using rule (15).

6 Discussion

We have contributed to further proliferation of logic-based ideas to the study of continuous and hybrid systems. Temporal logic offers a complementary way to evaluate real-valued signals, a way which is different from other commonly-used norms, measures and metrics, most of which are either point-wise or based on summation and averaging. We strongly believe that the measures introduced in this paper will find their application niche in situations where the interaction between timing and magnitude is non trivial as is the case in the design of electronic switching circuits or the analysis of biochemical pathways.

Future work includes the application of these measures and algorithms to the exploration of the parameter space of examples coming from the above application domains, including the incorporation of the sensitivity measure into gradient-based optimization procedures. To make the exploration procedure more effective, we intend to augment these measures with a more refined *diagnostics* to indicate more precisely what (Boolean combinations of) changes in the primary signals are required or sufficient in order to secure satisfaction. To this end we will need to propagate the information down from the secondary to primary signals and resolve possible conflicts due to the fact that the same primary signal may appear in several predicates. Finally it might be interesting

to see how the basic constructs of the calculus behave in the presence of the $\min$ and $\max$ operations.

References

1. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. R. Alur, T. Feder, and T.A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996.
3. M. Antoniotti, A. Policriti, N. Ugel, and B. Mishra. Model building and model checking for biochemical processes. *Cell Biochem Biophys*, 38(3):271–86, 2003.
4. E. Asarin, P. Caspi, and O. Maler. Timed regular expressions. *J. ACM*, 49(2):172–206, 2002.
5. S. Bensalem and D. Peled, editors. *Runtime Verification*, volume 5779 of *LNCS*, 2009.
6. L. Calzone, F. Fages, and S. Soliman. Biocham: an environment for modeling biological systems and formalizing experimental knowledge. *Bioinformatics (Oxford, England)*, 22(14):1805, 2006.
7. A. Donzé, B. Krogh, and A. Rajhans. Parameter synthesis for hybrid systems with an application to simulink models. In *HSCC*, LNCS. Springer-Verlag, April 2009.
8. A. Donzé and O. Maler. Systematic simulations using sensitivity analysis. In *HSCC'07*, LNCS, April 2007.
9. C. Eisner, D. Fisman, J. Havlicek, Y. Lustig, A. McIsaac, and D. Van Campenhout. Reasoning with temporal logic on truncated paths. In *CAV*, pages 27–39, 2003.
10. F. Fages and A. Rizk. From model-checking to temporal logic constraint solving. In *CP*, pages 319–334, 2009.
11. G.E. Fainekos and G.J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009.
12. Georgios E. Fainekos and George J. Pappas. *A User Guide for TaLiRo V0.1*, 2009.
13. Y. Kesten and A. Pnueli. A compositional approach to CTL* verification. *Theor. Comput. Sci.*, 331(2-3):397–428, 2005.
14. R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Syst.*, 2(4):255–299, 1990.
15. O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *FORMATS/FTRTFT*, pages 152–166, 2004.
16. O. Maler, D. Nickovic, and A. Pnueli. From MITL to timed automata. In *FORMATS*, pages 274–289, 2006.
17. O. Maler, D. Nickovic, and A. Pnueli. Checking temporal properties of discrete, timed and continuous behaviors. In *Pillars of Computer Science*, pages 475–505, 2008.
18. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag New York, 1991.
19. D. Nickovic and O. Maler. AMT: A property-based monitoring tool for analog systems. In *FORMATS*, pages 304–319, 2007.
20. A. Pnueli. The temporal logic of programs. In *Proc. 18th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 46–57, 1977.
21. A. Pnueli and A. Zaks. On the merits of temporal testers. In *25 Years of Model Checking*, pages 172–195, 2008.
22. A. Rizk, G. Batt, F. Fages, and S. Soliman. On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In *CMSB*, pages 251–268, 2008.
23. A. Rizk, G. Batt, F. Fages, and S. Soliman. A general computational method for robustness analysis with applications to synthetic gene networks. *Bioinformatics*, 25(12):i169–78, 2009.