On Expressing and Monitoring Oscillatory Dynamics

Petr Dluhoš, Luboš Brim, and David Šafránek *
Faculty of Informatics
Masaryk University
Botanická 68a, Brno, Czech Republic
safranek@fi.muni.cz

To express temporal properties of dense-time real-valued signals, the Signal Temporal Logic (STL) has been defined by Maler et al. The work presented a monitoring algorithm deciding the satisfiability of STL formulae on finite discrete samples of continuous signals. The logic has been used to express and analyse biological systems, but it is not expressive enough to sufficiently distinguish oscillatory properties important in biology. In this paper we define the extended logic STL* in which STL is augmented with a signal-value freezing operator allowing us to express (and distinguish) detailed properties of biological oscillations. The logic is supported by a monitoring algorithm prototyped in Matlab. The monitoring procedure of STL* is evaluated on a biologically-relevant case study.

1 Introduction

In this paper we deal with automatic decision of the question if a given continuous signal satisfies a given temporal property. This question originally arose in the domain of analogous circuits verification [17]. The procedure deciding this question has been called *monitoring* [17]. A monitor is constructed for a given property allowing off-line or even on-line decision over any continuous signal arising from a technical device (real case) or a numerical simulation procedure (model case) [18]. An important fact is that the monitoring procedure is always time bounded and that validity of the given logic property (discrete nature) is decided only approximately on a signal (continuous nature). These restrictions are necessary but not limiting to significantly help in avoiding errors in systems construction [21].

In systems biology, continuous signals most typically represent the model case – theoretical behaviour of mathematical models mimicking the dynamics of biological processes. Temporal properties are employed to express biological hypothesis [19, 9, 5, 4] and the monitoring procedure provides a promising analysis tool [7, 22, 11]. Many dynamical phenomena that arise in biological systems have the form of oscillations. In physics the term oscillation represents an infinite behaviour periodically alternating certain quantities/qualities. The identifying aspect of oscillations is the fact that certain states in the phase space of the dynamical system are being repeatedly re-visited. A biological example of such phenomena are circadian rhythms. In [23] an interesting wet-lab experiment has been achieved on the cyanobacterium *Cyanothece sp.* that showed a relation between metabolic cycles and circadian rhythms. A mathematical model of oscillations in gene regulation of cyanobacteria has been provided in [20]. Other models targeting oscillatory behaviour can be found e.g. in [15, 14].

In contrast to physics, the notion of oscillation in biology is usually understood informally and in a wider scope. Many wet-lab experiments show oscillatory behaviour with decreasing amplitude (so-called *damped oscillation*). Dual notion, oscillation with increasing amplitude, is also relevant: the question targeting how many oscillations and how strongly they increase until a permanent oscillation is

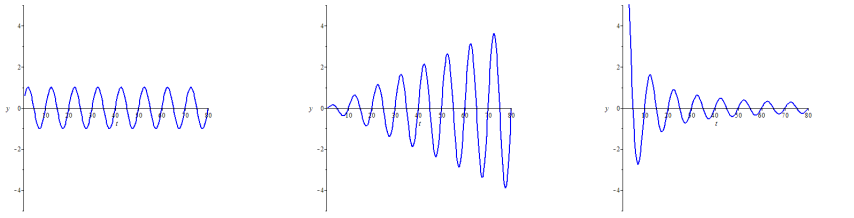
*The work has been supported by the Grant Agency of Czech Republic grant GAP202/11/0312.

Ezio Bartocci and Luca Bortolussi (Eds.): HSB 2012
EPTCS 92, 2012, pp. 73–87, doi:10.4204/EPTCS.92.6

achieved comes to a significant interest when tuning biological processes via mathematical models [15, 14]. When considering the population behaviour as a real-valued signal in dense-time domain, we need to quantitatively express and compactly encode the mentioned types of oscillatory phenomena.

In [9], a linear-time logic with constraints over real values has been defined. It can be practically used to express many dynamical phenomena including oscillations of species concentration. The logic is interpreted over finite time-series of real-valued data obtained as discretely sampled solutions of differential equations (ODEs) (or a series of physical measurements). Formally, a technical problem may arise with the dense-time domain of considered signals. In general, in a particular interval on the time axis of the signal there may exist infinitely many points where a considered property (a predicate over real-values) changes its truth value. A rigorous semantics which treats this problem is given in [17] in terms of continuous (real-valued) signals which are required to be piece-wise affine. In such a setting, it is assumed that there exists a finite interval covering of the signal time-domain where the end points can be sampled and on which the signal behaves "reasonably" in each individual time interval. The logic with such a semantics is called *Signal Temporal Logic (STL)* [17] and is based on the (bounded) Metric Interval Temporal Logic (MITL) [2].

Neither STL, nor other temporal logic, as far as we know, provide any possibility to express (and distinguish) the classes of oscillations such as damped oscillations or oscillations with increasing amplitude. The reason is impossibility of globally referencing (and relatively comparing) concrete signal values occurring in time points in which some local property is satisfied. Of course, in STL we can express directly any single concrete signal, but not a universal property without references to concrete values. E.g., the damped oscillation in Figure 1c can be expressed in STL as a sequence exactly reaching the 15 local extremes in the given order. In general, there can appear any number of local extremes in the observed time interval. Such a general property cannot be expressed in STL.



(a) $x(t) = sin(\frac{2\pi t}{10})$ (b) $x(t) = \frac{t}{20} sin(\frac{2\pi t}{10})$ (c) $x(t) = \frac{20}{t} sin(\frac{2\pi t}{10})$

Figure 1: Various types of oscillations.

In this paper, we propose an extension of STL, denoted STL*, based on enriching the logic with a *signal-value freeze operator* $*[\varphi]$ which allows referencing to a signal value in a time point when φ is determined true. By means of atomic propositions enriched by variables of the form x^* referring to "frozen" values, we can express damped oscillations by the following formula:

$$\mathbf{G}_{[0,60]}\,((\mathbf{F}_{[0,10]} * \big[\mathbf{G}_{[1,10]}\, x^* \geq x + c\big]) \wedge (\mathbf{F}_{[0,10]} * \big[\mathbf{G}_{[1,10]}\, x^* \leq x - c\big])).$$

It says that *"there is always a time instant in near future which is a local maximum for some future interval and there is also another time instant in near future which is a local minimum for some future interval"*. In these intervals, another local maxima and minima for more distant intervals occur, and so on. This implies that the signal values can be surrounded by a decreasing function from top and an increasing function from bottom. The constant c sets the extent of damping.

The concept of freeze quantification has been introduced in [3] to increase real-time temporal logic expressiveness by allowing every temporal operator to bind a time variable to the time it refers to. In the context of systems biology this concept has been used in Biological Oscillator Synchronisation Logic (BOSL) [6] to reason on oscillators synchronisation. In our work we shift the notion of temporal quantification to bind the real-valued signal variables to the temporal reference of time and we do it at the level of STL logic. Freezing of variable values has been used before in higher-level formalisms employing local variables [13, 1]. However, these formalisms do not support dense-time binding.

It is necessary to note that when values of derivatives of signal values are known in all sampled points, some oscillatory properties can be expressed by using plain STL with predicates over derivatives. However, in such a case all needed derivatives (of all required orders) must be included within the signal. Getting these data can be computationally hard or even impossible. STL* allows to express (and monitor) oscillatory properties without the need of any additional information supporting the signals.

The contribution of this paper is an extension of STL logic with signal value freeze quantification (Section 3) motivated by the need to express oscillations appearing in biology. The paper is primarily focused on a practical aspect, we give an algorithm for monitoring STL* formulae over (bounded) continuous signals (Section 4). The result is supported by a prototype implementation evaluated on a biological case study of *E. coli* repressilator (Section 5). Please note that a detailed discussion of STL* expressiveness is provided in a master's thesis [10] that makes a preliminary version of this paper.

2 Background

First, we briefly recall the real-valued signals over continuous time from [17]. For the purposes of this paper, we focus only on signals over $\mathbb{R}^n$.

Definition 2.1 *Let $T = [0,r], r \in \mathbb{Q}_{\geq 0}, T \subseteq \mathbb{R}_{\geq 0}$, be a finite interval (*time domain *of the signal). A* finite length signal *s is a function $s : T \to \mathbb{R}^n$ where $|s| = r$ denotes the* length of the signal *s.*

Definition 2.2 *Given a signal $s : T \to \mathbb{R}^n$, n denotes the* order of the signal *s. We use s_i to denote the i-th component of the signal, $s_i : T \to \mathbb{R}$ (i.e., the canonical projection to the i-th element).*

A signal of order n represents one finite run of a continuous-time system with n variables. A variable can represent any quantity of the system, e.g., concentration of species, its derivative or any other attribute of the signal which can be measured and quantified.

To translate the properties of a signal into terms of logic, we use a function transforming real values of the signal components in every time instant to the Boolean set $\{0,1\}$, representing satisfaction or dissatisfaction of a property over the signal. These truth values are used as propositional variables in formulae of the logic. Unlike in [17], we restrict the allowed operations on signal variables to linear combinations and simple comparisons. Such a restriction is needed for our monitoring algorithm (Section 4) and does not limit expression of the properties of our interest.

Definition 2.3 Linear predicate ν of order m *is a function $\nu : \mathbb{R}^m \to \mathbb{B}$ of the form $\sum_{i=1}^{m} a_i x_i \sim b$ where $a_i, b \in \mathbb{R}$ are real coefficients, x_i are input variables, $\sim \in \{<, \leq, >, \geq, =\}$ and $\mathbb{B}$ is a Boolean set $\mathbb{B} = \{0,1\}$. The predicate $\nu(x_1, x_2, \ldots x_m)$ returns 1 iff the expression $\sum_{i=1}^{m} a_i x_i \sim b$ is true and 0 otherwise.*

However, in the logic defined in this paper, we need not only to work with the values of variables acquired in a single time instant, but also with a second set of values from another time instant. For this reason, we will define an additional function extracting truth values from the signal.

Definition 2.4 Atomic predicate μ *is a function* $\mu : \mathscr{S}_n \times \mathbb{R}_0^+ \times \mathbb{R}_0^+ \to \mathbb{B}$ *where* $\mathscr{S}_n$ *is a class of signals of order n. For each signal* $s \in \mathscr{S}_n$ *and* $t, t^* \in [0, |s|]$, μ *is defined by the formula:*

$$\mu(s,t,t^*) \stackrel{df}{=} \nu(s_1(t), \ldots, s_n(t), s_1(t^*), \ldots, s_n(t^*)),$$

where ν *is a linear predicate of order* $2n$ *and* s_i *are components of the signal s. Values of the atomic predicate* μ *for* $t > |s|$ *or* $t^* > |s|$ *are considered undefined.*

An atomic predicate over a signal s describes a Boolean property of the signal s, i.e., if a constraint specified by the linear predicate ν is satisfied with respect to two time instants t and t^*. An example of such a property is *"under condition* $t^* < t$*, the difference between values of variables* s_1 *and* s_2 *was higher in the time* t^* *than in the time* t*"*:

$$\mu(s,t,t^*) = \nu(s_1(t), s_2(t), s_1(t^*), s_2(t^*)) = s_1(t) - s_2(t) < s_1(t^*) - s_2(t^*),$$

which can be rearranged to the standard form of a linear predicate according to Definition (2.3):

$$\nu(s_1(t), s_2(t), s_1(t^*), s_2(t^*)) \stackrel{\text{df}}{=} s_1(t) - s_2(t) - s_1(t^*) + s_2(t^*) < 0.$$

Instead of $\mu(s,t,t^*) = s_1(t) - s_2(t) < s_1(t^*) - s_2(t^*)$ we usually write $\mu(s,t,t^*) = x - y < x^* - y^*$, denoting the variables of a signal s in time t by letters $x, y, z, \ldots$ and the same variables in time t^* as $x^*, y^*, z^*, \ldots$.

As in [17], we will avoid the undesired cases of atomic predicates where the output values of the predicate are varying infinitely often. We assume that we deal with signals that are well-behaving with respect to every μ, that is, $\mu(s)$ has a bounded variability and every change in $\mu(s)$ is detected in the sense that every point t such that $\mu(s[t]) \neq lim_{t' -> t} \mu(s[t'])$ is included in the sampling [17].

3 A Logic for Oscillatory Dynamics

In this section we describe the syntax and semantics of STL*. It is based on the Signal Temporal Logic (STL) [17] which we extend with the signal-value freeze operator $*[\varphi]$. We consider atomic predicates restricted to linear as defined in the previous section.

3.1 Syntax

The formulae of the STL* are inductively defined by the following grammar:

$$\varphi ::= \mu_i \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \, \mathbf{U}_{[a,b]} \, \varphi_2 \mid *\varphi,$$

where μ_i are atomic predicates from Definition 2.4, $\neg$ and $\vee$ are standard propositional logic operators, $\mathbf{U}_{[a,b]}$ is a bounded until operator constrained by the closed nonsingular time interval $[a,b]$ with rational end-points. The interpretation of this operator is similar to the one used in the Metric Interval Temporal Logic (MITL) [2]. Finally, $*$ is the newly added unary *signal-value freeze operator*. In the standard way, we can derive additional temporal operators such as eventually ($\mathbf{F}_{[a,b]} \, \varphi \equiv \mathsf{true} \, \mathbf{U}_{[a,b]} \, \varphi$) and globally ($\mathbf{G}_{[a,b]} \, \varphi \equiv \neg \mathbf{F}_{[a,b]} \neg\varphi$).

3.2 Semantics

Formulae of STL* are interpreted over triplets (s,t,t^*) where s is a real-valued signal and $t,t^* \in [0,|s|]$ are two time instants. We write $(s,t,t^*) \models \varphi$, iff a formula φ is satisfied on the triplet (s,t,t^*). The satisfaction of an STL* formula is defined inductively to its structure:

$$\begin{array}{lcl}
(s,t,t^*) \models \mu & \overset{\text{df}}{\iff} & \mu(s,t,t^*) = 1; \\
(s,t,t^*) \models \neg\varphi & \overset{\text{df}}{\iff} & (s,t,t^*) \not\models \varphi; \\
(s,t,t^*) \models \varphi_1 \vee \varphi_2 & \overset{\text{df}}{\iff} & (s,t,t^*) \models \varphi_1 \text{ or } (s,t,t^*) \models \varphi_2; \\
(s,t,t^*) \models \varphi_1 \mathbf{U}_{[a,b]} \varphi_2 & \overset{\text{df}}{\iff} & \exists t' \in [a+t,b+t] : (s,t',t^*) \models \varphi_2 \wedge \\
 & & \forall t'' \in [t,t'] : (s,t'',t^*) \models \varphi_1; \\
(s,t,t^*) \models *\varphi & \overset{\text{df}}{\iff} & (s,t,t) \models \varphi.
\end{array} \tag{1}$$

Because models for STL* are signals, we also speak about satisfaction of a formula over a signal.

Definition 3.1 *The satisfaction of a formula φ over a signal s is defined as: $s \models \varphi \overset{df}{\iff} (s,0,0) \models \varphi$.*

The meaning of the operator $*\varphi$ is to freeze the values of the signal over which the formula is interpreted in the current time point so we can use these values inside the subformula φ. Formula $*\varphi$ is true in time $t = t_0$ iff the formula φ is true with the time t^* frozen in this time, i.e., $t^* = t_0$. In combination with other temporal operators, interesting properties can be expressed such as *"the value of the variable x is nondecreasing on the interval I"*, $\varphi_1 \equiv \mathbf{G}_I *\big[\mathbf{G}_{[0,\varepsilon]}\,(x^* \le x)\big]$, where $\varepsilon > 0$ is an arbitrary constant. Or *"at some point in the future (during interval I) the value x increases by the value of 5 within two time units"*, $\varphi_2 \equiv \mathbf{F}_I *\big[\mathbf{F}_{[0,2]}\,(x^* + 5 = x)\big]$.

To determine the satisfaction of a formula over a signal, the signal has to be of a sufficient length. The necessary length can be computed for each formula inductively to its structure as in [17]. The freeze operator does not require any additional length:

$$\begin{array}{lcl}
l(\mu) & \overset{\text{df}}{=} & 0; \\
l(\neg\varphi) & \overset{\text{df}}{=} & l(\varphi); \\
l(\varphi_1 \vee \varphi_2) & \overset{\text{df}}{=} & \max(l(\varphi_1), l(\varphi_2)); \\
l(\varphi_1 \mathbf{U}_{[a,b]} \varphi_2) & \overset{\text{df}}{=} & \max(l(\varphi_1), l(\varphi_2)) + b; \\
l(*\varphi) & \overset{\text{df}}{=} & l(\varphi).
\end{array} \tag{2}$$

From Definition 3.1 we can see that if a formula φ contains an atomic predicate μ_i and there is no operator $*$ wrapping this predicate then all variables concerning the frozen time instant t^* are handled as if $t^* = 0$.

From the definition of the semantics of STL* (1) follows that in a formula with several nested operators $*$, the meaning of a frozen variable x^* is local. I.e., it relates to the nearest operator $*$, because the content of the variable t^* is overwritten by the most nested freeze operator $*$. It implies that only a single set of frozen signal values is accessible at every place in the formula. For example, in a formula:

$$*\big[x^* \le y\, \mathbf{U}_I\, (x^* = y \wedge *\big[\mathbf{G}_{[0,5]}\, x^* \ge y\big])\big], \tag{3}$$

the first and the second occurrence of the variable x^* relate to the first usage of $*$ and address the time $t = 0$, in which the formula is interpreted. The third occurrence of x^*, in contrast to the previous two,

relates to the second operator $*$ and addresses the time instant in which the subformula on the right side of $\mathbf{U}_I$ becomes true. Semantics restricted in this way does not excessively limit the capability of expressing various biological behaviour while remaining computationally feasible.

4 Algorithm

In this section we deal with monitoring of an STL* formula, which means to determine the satisfaction of a formula over a finite length signal. The monitoring procedure introduced in this paper is inspired by constrained LTL model-checking [9] and monitoring of STL formulae [17], but needed to be extended into 2D space due to freeze operator $*$. It is because the task is solved simultaneously for two time instants, actual time and frozen time.

The idea of the monitoring procedure is to construct a parse tree of the formula and check the satisfaction in a bottom-up manner. First step in checking a formula φ over a signal s is to construct a set of time points in which the formula φ is satisfied.

Definition 4.1 *Let φ be a STL* formula and s be a real-valued signal. A set $S_{\varphi,s} = \{(t,t^*) \in [0,|s|] \times [0,|s|] \mid (s,t,t^*) \models \varphi\}$ is called the* satisfaction set *of formula φ over signal s. We use short-form notation S_φ whenever the signal s is obvious from the context.*

Now we inductively construct satisfaction sets for nodes on higher levels of the parse tree. The last step of the procedure is to decide if $s \models \varphi$ from satisfaction set for the formula φ. According to Definition 3.1, it is equivalent to checking whether the satisfaction set contains the point $(0,0)$.

Constructing the satisfaction set for a general signal can be a very difficult task. For this reason we will consider the monitoring algorithm only for piecewise linear signals. This is a reasonable requirement because in most cases we deal with time series produced by numerical simulations of modelled systems or by some measurements. These series can be interpreted as piecewise linear signals considering the values changing linearly between two adjacent points. If required, other points can be generated in between the existing points to make the signal more precise.

Definition 4.2 *A real-valued signal s of order n is called* piecewise linear signal *iff all the projections $s_i(t) : [0,|s|] \to \mathbb{R}, i = 1,\ldots,n$ are piecewise linear functions defined on a finite set of intervals $\mathscr{J} = \{I_1, I_2, \ldots, I_m\}$ where $I_j \subseteq [0,|s|], j = 1,\ldots,m$ and $\bigcap \mathscr{J} = \emptyset, \bigcup \mathscr{J} = [0,|s|]$.*

Theorem 4.1 (Inductive construction of the satisfaction set) *Let s be a piecewise linear signal and φ a formula in STL*. The satisfaction set S_φ of the formula φ over the signal s can be constructed inductively with respect to the structure of the formula φ:*

$$
\begin{array}{lcl}
S_\mu & = & \{(t,t^*) \in \mathbb{R}_0^+ \times \mathbb{R}_0^+ \mid \mu(s,t,t^*) = 1\}; \\
S_{\neg\varphi} & = & \mathbb{R}_0^+ \times \mathbb{R}_0^+ \setminus S_\varphi; \\
S_{\varphi_1 \vee \varphi_2} & = & S_{\varphi_1} \cup S_{\varphi_2}; \\
S_{\varphi_1 \mathbf{U}_{[a,b]} \varphi_2} & = & \{(t,t^*) \in S_{\varphi_1} \mid \exists t' \in [t+a, t+b] : (t',t^*) \in S_{\varphi_2} \wedge \\
 & & \forall t'' \in [t,t'] : (t'',t^*) \in S_{\varphi_1}\}; \\
S_{*\varphi} & = & \{(t,t^*) \in \mathbb{R}_0^+ \times \mathbb{R}_0^+ \mid (t,t) \in S_\varphi, t^* \in \mathbb{R}_0^+\}.
\end{array}
$$

Proof *Each of the equations follows directly from the definition of semantics of STL** (1).

The assumption of piecewise linear signals in combination with linearity of atomic predicates (Definitions 2.3, 2.4) enables us to construct satisfaction sets for atomic predicates in polynomial time and to easily compute sets belonging to higher formulae. The formalism we use for the representation of satisfaction sets are convex polytopes.

Definition 4.3 Convex polytope $P \subseteq \mathbb{R}^n$ *is a set* $\{x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^{n\times 1} \mid Ax \leq b\}$ *where* $A \in \mathbb{R}^{m\times n}$ *is a matrix and* $b \in \mathbb{R}^{m\times 1}$ *is a column. A convex polytope in* $\mathbb{R}^1$ *is called a* line segment*, in* $\mathbb{R}^2$ *a* convex polygon*.*

Each inequality $\sum_{j=1}^{n} a_{ij}x_j \leq b_i$ in Definition 4.3 identifies a subspace of $\mathbb{R}^n$. The convex polytope is obtained as intersection of these subspaces.

Theorem 4.2 *Assume s a piecewise linear signal and* $\mathscr{J} = \{I_1, \ldots, I_m\}$ *the set of intervals on which s is defined. Let* μ *be an atomic predicate over the signal s. The predicate* $\mu(s,t,t^*) = \sum_{i=1}^{n} a_i s_i(t) + \sum_{i=1}^{n} b_i s_i(t^*) \sim b, \sim \in \{<, \leq, >, \geq, =\}$ *is a linear function over the signal s in variables t and* t^* *according to Definition 2.4. Denote* $S^+_{i,j} \stackrel{df}{=} \{(t,t^*) \in I_i \times I_j \mid \mu(s,t,t^*) = 1\}$ *where* $i, j \in \{1, \ldots m\}, I_i, I_j \in \mathscr{J}$ *the set of time instants at which the atomic predicate* μ *is satisfied over signal s on rectangular area* $I_i \times I_j$ *and* $S^-_{i,j} \stackrel{df}{=} (I_i \times I_j) \setminus S^+_{i,j}$ *its complement. For* $t \in I_i, t^* \in I_j$ *there can arrive two situations:*

1. *if* $\sim \equiv =$*, then the set* $S^+_{i,j}$ *makes a line segment (possibly degenerated to a single point or an empty set);*
2. *if* $\sim \not\equiv =$*, then the space* $I_i \times I_j$ *is divided by the line* $\sum_{i=1}^{n} a_i s_i(t) + \sum_{i=1}^{n} b_i s_i(t^*) = b$ *into two subspaces* $S^+_{i,j}$ *and* $S^-_{i,j}$ *(Figure 2). (Again, there might be a degenerated case when the line* $\sum_{i=1}^{n} a_i s_i(t) + \sum_{i=1}^{n} b_i s_i(t^*) = b$ *does not cross the rectangle* $\{(t,t^*) \in I_i \times I_j\}$*. In this case* $S^+_{i,j} = \emptyset$ *and* $S^-_{i,j} = I_i \times I_j$ *or vice versa).*

Proof *Described properties result from the linear form of atomic predicates 2.4 and from the fact that signal s behaves linearly on each interval* $I_i \in \mathscr{J}, i \in \{1, \ldots m\}$.

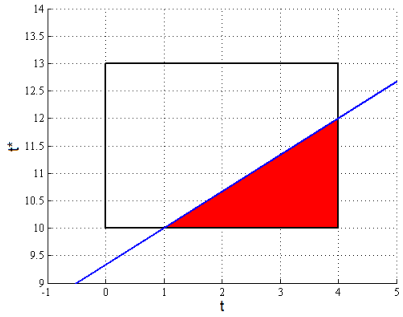


Figure 2: An illustration of the set $S^+_{i,j}$ for a formula $x^* < \frac{2}{3}x + \frac{28}{3}$ on the subspace $I_i \times I_j = [0,4] \times [10,13]$.

In both cases, the set $S^+_{i,j}$ makes a convex set easily describable by set operations over convex polytopes. In the first case, a line segment itself is a convex polytope. In the second case, either the set $S^+_{i,j}$ is a convex polytope, or (when $\sim \in \{<, >\}$) it is a convex polytope after subtraction of one bordering line segment, which is a polytope.

Using this algorithm we can express the satisfaction set S_μ as $S_\mu = \bigcup_{i,j \in \{1,\ldots,m\}} S^+_{i,j}$, where $S^+_{i,j}$ are the sets described above constructed by set operations on polytopes. The set S_μ can be further simplified by

joining adjacent polytopes. Having satisfaction sets for atomic predicates, we could inductively construct satisfaction sets for composite formulae from Theorem 4.1 again as operations on polytopes, which could be achieved in polynomial time [8].

However, working with precise satisfaction sets as defined in Theorem 4.1 can be quite a difficult task. We have to work with polytopes of different dimensions (polygons, lines and points) and operations over sets of these objects can be unnecessarily demanding. In real-world applications, we could end up performing an expensive monitoring over noisy signals measured on imperfect devices or computed on computers with finite numerical precision. Hence think about less precise, yet faster monitoring algorithm.

We can imagine a *noisy signal* as a signal measured or computed with finite precision. The actual values of the variables can differ from the values presented by the signal up to some error $\varepsilon > 0$. This inaccuracy in the domain of values implies also inaccuracy in the time domain. When investigating the time instant in which some event occurred (e.g., $x > 0$) we cannot be sure when exactly it happened. That is because the critical value $x = 0$ is burdened by an error and it is possible that $x = 0$, but also that $x > 0$ or $x < 0$ in that particular time instant.

Imagine we are in the situation from Theorem 4.2, but working with a noisy signal. We cannot be sure about the border of the set $S^+_{i,j}$ due to the error ε. The condition $\sum_{i=1}^{n} a_i s_i(t) + \sum_{i=1}^{n} b_i s_i(t^*) = b$ defining points on the border of $S^+_{i,j}$ could be easily broken by changing the values of the signal s by arbitrary small values from $(0, \varepsilon)$. For this reason, we will ignore these border points and count them as they were in $S^+_{i,j}$ or in $S^-_{i,j}$ depending on what would be computationally easier. As a consequence, in the first case in Theorem 4.2, $S^+_{i,j} = \emptyset$, and in the second case, it does not matter if $\sim \equiv <$ or $\sim \equiv \leq$.

As a result of this simplification, the set of allowed operators in atomic predicates is restricted to $\{<, >\}$. The remaining three operators lost their sense. With signals burden with some error $\varepsilon > 0$ the satisfaction of atomic predicates can be meaningfully determined only for time instants inside the satisfaction set. Hence the satisfaction of a formula using the operator $=$, e.g., $x = b$ or $x = y + 2$, cannot be determined. Every reference to a precise value has to be replaced by a sufficiently large interval, e.g., formula $x = b$ can be replaced by $x \geq (b - \delta) \wedge x \leq (b + \delta)$ for some $\delta > \varepsilon/2$.

Based on these assumptions, monitoring can be solved approximately. For every pair of intervals $I_i, I_j \in \mathscr{J}$, the satisfaction set on this area can be described by a single convex polygon $S^+_{i,j}$. For the whole satisfaction set $S_{\mu,s}$ we get $S_{\mu,s} = \bigcup_{i,j \in \{1,\ldots,m\}} S^+_{i,j}$. The problem of construction of satisfaction sets for composite formulae from Theorem 4.1 is reduced to operations with sets of convex polytopes in plane for which efficient algorithms exist [8].

Algorithm 4.1 (Approximative monitoring algorithm for piecewise linear signals)

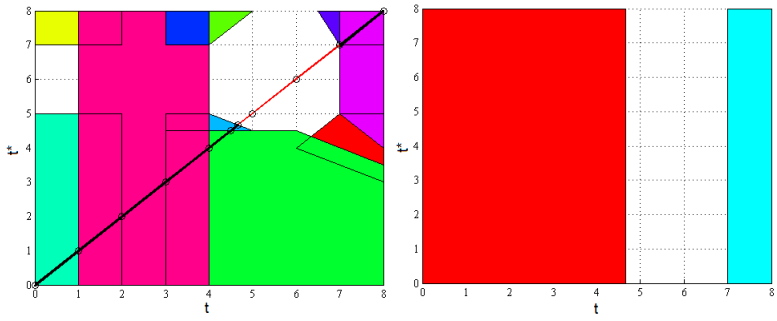
Input: A piecewise linear signal s and an STL* formula φ.
Output: Answer to the question $s \models \varphi$.

Algorithm inductively constructs the satisfaction set S_φ:

All the computations are performed on parts of the signal s of sufficient length. This can be computed according to (2). If the signal does not have the sufficient length, the answer returned by the algorithm might be wrong.

1. $S_\mu = \bigcup_{i,j \in \{1,\ldots,m\}} S^+_{i,j}$. The computation of the satisfaction sets $S^+_{i,j}$ for individual intervals is performed according to Theorem 4.2, but without determining the satisfaction of the borders (as was justified in this section).

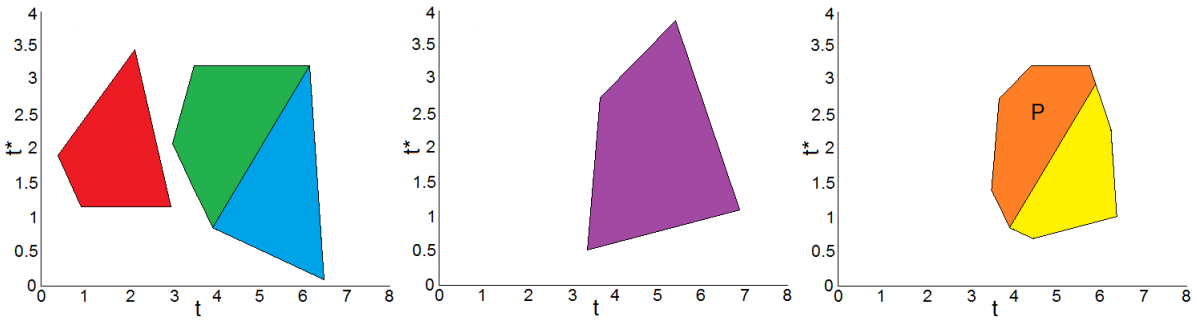
2. $S_{\neg\varphi} = (\mathscr{J} \times \mathscr{J}) \setminus S_\varphi$. The result can be computed as a simple Boolean operation on polygons. It is necessary to ensure that the resulting set consists only of convex polygons, which could be achieved for example by triangulation [8].
3. $S_{\varphi_1 \vee \varphi_2} = S_{\varphi_1} \cup S_{\varphi_2}$. Again a Boolean polygonal operation.
4. $S_{*\varphi} = \{(t,t^*) \in \mathbb{R}_0^+ \times \mathbb{R}_0^+ \mid (t,t) \in S_\varphi, t^* \in \mathbb{R}_0^+\}$. The satisfaction set for the freeze operation can be computed by: (1) finding an intersection between the line $t^* = t$ and the satisfaction set S_φ (black line segments in Figure 3a), (2) substituting values in the second component by the whole axis t^*, i.e., making a projection to the first component t and then a Cartesian product with the axis t^* (Figure 3b).



(a) (b)

Figure 3: Example of computing satisfaction set for the formula $*\varphi$. a) The satisfaction set represented as a set of (overlapping) convex polygons (different colors are used only for depicting the fact that the set consists of convex polygons) and the line $t = t^*$. The regions of intersection of the line and the satisfaction set are depicted in black color. b) Resulting set $S_{*\varphi}$ obtained by projecting the intervals of intersection to the axis t and making a Cartesian product with the axis t^*.

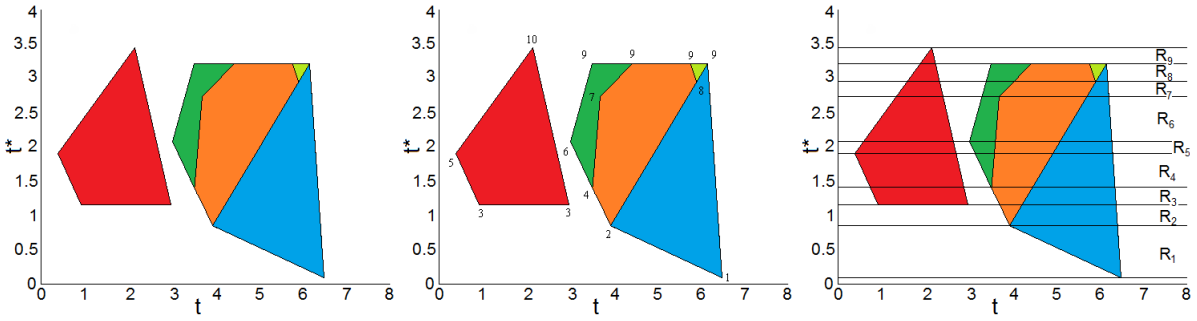
5. $S_{\varphi_1 \mathbf{U}_{[a,b]} \varphi_2} = \{(t,t^*) \in S_{\varphi_1} \mid \exists t' \in [t+a, t+b] : (t',t^*) \in S_{\varphi_2} \wedge \forall t'' \in [t,t'] : (t'',t^*) \in S_{\varphi_1}\}$. To be more illustrative, we explain this part of the algorithm from the geometrical point of view, identifying values t with the horizontal axis and values t^* with the vertical axis.



(a) S_{φ_1} (b) S_{φ_2} (c) $S_{\varphi_1} \cap S_{\varphi_2}$

Figure 4: First step of computation of $S_{\varphi_1 \mathbf{U}_{[a,b]} \varphi_2}$ is to find the intersection $S_{\varphi_1} \cap S_{\varphi_2}$.

For each convex polygon $P \in S_{\varphi_1} \cap S_{\varphi_2}$ (Figure 4c) the following procedure is performed:



(a) $P \cup S_{\varphi_1}$ (b) ordered vertices (c) rectangular polygons (stripes) $R_i, i = 1, \ldots, n-1$

Figure 5: Next, the space is divided into horizontal stripes. The task is solved in each stripe separately.

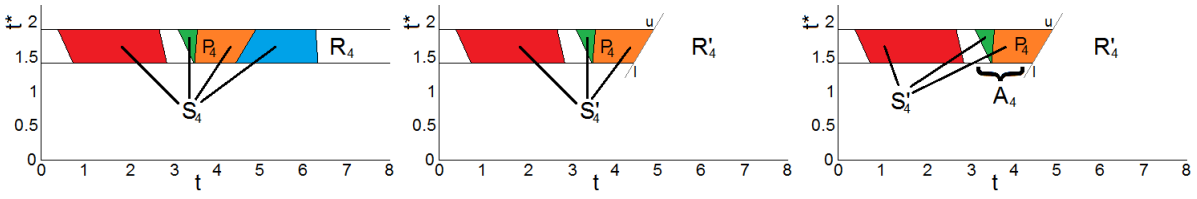
(a) Vertices of polygons $\{P\} \cup S_{\varphi_1}$ are increasingly ordered by the second component t^* (duplicates can be removed). Denote the ordered set $V \stackrel{\text{df}}{=} (V_1, V_2, \ldots V_n)$ (Figures 5a, 5b);

(b) For $i = 1, \ldots, n-1$ every neighbouring pair of vertices $V_i = (t_i, t_i^*), V_{i+1} = (t_{i+1}, t_{i+1}^*) \in V$ specifies a rectangular polygon (stripe):

$$R_i \stackrel{\text{df}}{=} \{(t, t^*) \in \mathbb{R}_0^+ \times [t_i^*, t_{i+1}^*]\}$$

(Figure 5c). R_i in a form of line segment is considered empty because we do not care about the border points.

(c) The task is solved in every stripe R_i separately. For every nonempty R_i denote $P_i \stackrel{\text{df}}{=} R_i \cap P$ and $S_i \stackrel{\text{df}}{=} R_i \cap S_{\varphi_1}$. Due to the way of construction of R_i, all polygons in S_i and the single polygon P_i have all their vertices on the upper ($t^* = t_{i+1}^*$) or the lower ($t^* = t_i^*$) border of the stripe R_i (Figure 6a). In fact, these polygons can take only the shape of a triangle or a trapezoid (Figure 7).



(a) the stripe R_4 (b) points on the left side of the line $\overrightarrow{ul}$ (c) A_4 = the rightmost polygon in S'_4

Figure 6: The potential polygons are identified. (The solution cannot be placed further in time than the property φ_2. Moreover, the property φ_1 must continuously hold at all time points until φ_2 is satisfied.)

Now we get rid of the polygons lying on the right side of P_i because their points cannot be in the solution (they represent the time past the event φ_2). To do so, we isolate the upper and lower rightmost vertices of the polygon P_i. Denote them as $u = (u_t, u_{t^*})$ and $l = (l_t, l_{t^*})$ (Figure 6b). We get a new area:

$$R'_i \stackrel{\text{df}}{=} \{(t, t^*) \in R_i \mid (t, t^*) \text{ lies on the left side of the line } \overrightarrow{ul}\}.$$

The solution can lie only in this area, so we can restrict S_i to $S'_i \stackrel{\text{df}}{=} R'_i \cap S_i$.

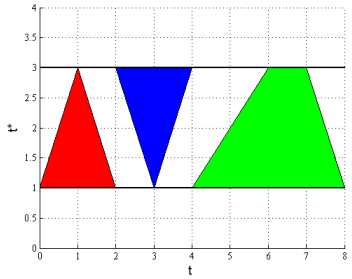


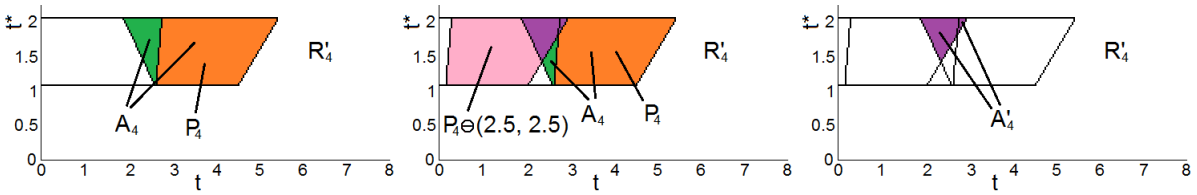
Figure 7: An example of a stripe and all possible shapes of polygons inside a stripe.

(d) All the adjoining polygons (sharing an edge) in S'_i are connected to form maximal seamless convex polygons (e.g., the green and orange polygons in Figure 6b).

(e) Let A_i be the rightmost polygon in S'_i (Figure 6c) (Even after connecting some polygons in S'_i their shape can still be only triangular or trapezoidal (Figure 7) so there is only one rightmost polygon in S'_i). Then $A'_i = A_i \cap (P_i \ominus (b,a))$ is the solution for the stripe R_i (Figure 8).

The operation $\ominus$ is defined as $A \ominus (a,b) \stackrel{\text{df}}{=} \{(x,y) \in \mathbb{R} \times \mathbb{R} \mid \exists c \in (a,b) : (x+c,y) \in A\}$ which is equal to $A \oplus (-a,-b)$, where $\oplus$ is the Minkowski sum.

(f) The final satisfaction set is given by $S_{\varphi_1 \mathbf{U}_{[a,b]} \varphi_2} = \bigcup_{i=1}^{n-1} A'_i$.



(a) P_4 before application of the operator $\ominus$ (b) the shifted polygon $P_4 \ominus (2.5,2.5)$ (c) $A'_4 = A_4 \cap (P_4 \ominus (2.5,2.5))$, the final solution in the stripe R'_4

Figure 8: Last step is to consider the interval $[a,b]$ bounding the operator $\mathbf{U}$.

6. The result $s \models \varphi$ which is equivalent to $(0,0) \in S_\varphi$ is returned.

Remark 4.1 *The satisfaction set for temporal operator future* $\mathbf{F}_{[a,b]}\,\varphi \equiv \mathsf{true}\,\mathbf{U}_{[a,b]}\,\varphi$ *(and hence* $\mathbf{G}_{[a,b]}\,\varphi$*) can be computed more easily than according to the step 5. It can be obtained directly as* $S_{\mathbf{F}_{[a,b]}\,\varphi} = S_\varphi \ominus (b,a)$.

4.1 Time and space complexity

Time and space complexity of all steps of the Algorithm 4.1 is proportional to the number of polygons they are working with. Number of polygons generated in step 1 is at most n^2 for each atomic predicate, where n is the number of intervals on which the signal is defined.

The number of polygons does not asymptotically change during the computation of operations described in other steps of the algorithm and all these functions do not work slower than $O(n^2)$ (see [8] for more details). The output polygons also keep small number of vertices during the process.

The upper bound on the total computation time is therefore $O(kn^4)$, where n is the number of intervals on which the signal is defined and k is the size of the investigated formula.

5 Evaluation and Case Study

A prototype of the approximative monitoring Algorithm 4.1 was implemented in Matlab (version R2011b). The Multi-Parametric Toolbox (MPT) package [16] (version 2.6.3) was used for the polygonal operations. We have not focused on efficiency of the implemented algorithm, our goal was to prove the concept of the presented algorithm. Detailed description of the implementation, results of the experiments and performance analysis can be found in [10].

We have studied a biological system of three transcriptional repressors which was designed and built into bacteria *Escherichia coli* [14]. Concentrations of involved proteins are periodically oscillating which causes periodic production of a green fluorescent protein. Intensity of fluorescence of this protein, which can be measured, gives evidence of the ongoing activity of the network (Figure 9).

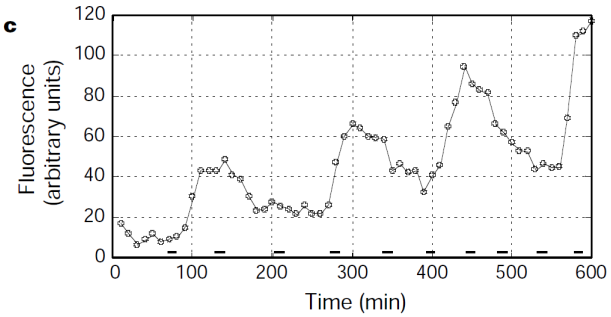


Figure 9: The fluorescent intensity of a colony of *Escherichia coli* containing the repressilator. The intensity is increasing overall due to the increasing number of individuals in the colony. The period of oscillations is lower than the time between cell division, but the whole repressilator is transmitted from generation to generation. Figure was taken from [14].

The network was modelled as a system of six differential equations:

$$\begin{array}{rclcrcl} \dot{m}_1 &=& -m_1 + \frac{\alpha}{1+p_3^n} + \alpha_0; & & \dot{p}_1 &=& -\beta(p_1 - m_1); \\ \dot{m}_2 &=& -m_2 + \frac{\alpha}{1+p_1^n} + \alpha_0; & & \dot{p}_2 &=& -\beta(p_2 - m_2); \\ \dot{m}_3 &=& -m_3 + \frac{\alpha}{1+p_2^n} + \alpha_0; & & \dot{p}_3 &=& -\beta(p_3 - m_3). \end{array} \tag{4}$$

Where $\dot{m}_i \stackrel{\text{df}}{=} \frac{dm_i}{dt}, \dot{p}_i \stackrel{\text{df}}{=} \frac{dp_i}{dt}$; m_1, m_2 and m_3 correspond to activity of genes $lacl, tetR$ and cl; p_1, p_2 and p_3 are concentrations of corresponding proteins and $\alpha, \beta, \alpha_0, n$ are constants (a typical behaviour of the system is depicted in Figure 10).

The analysis of the system (4) and the estimation of parameters $\alpha, \beta, \alpha_0, n$ producing the oscillatory behaviour was originally done by means of manual qualitative analysis of ODEs [14]. However, we can perform the analysis automatically using the monitoring of STL* formulae. We can specify the desired oscillatory property by the formula:

$$\varphi \stackrel{\text{df}}{=} \mathbf{G}_{[10,190]}\, \mathbf{F}_{[0,50]} * \big[(\mathbf{F}_{[1,50]}\, m_i^* < m_i) \wedge \mathbf{F}_{[1,50]}\, m_i^* > m_i) \big] \tag{5}$$

and test the satisfaction for different values of parameters on runs of the length at least 300 minutes. The expected period has to be lower than 50 minutes. We start the testing 10 minutes later after the beginning to avoid the initial swing and end the testing 50 minutes before the end of the measurement because the signal might be cut in the middle of a period.

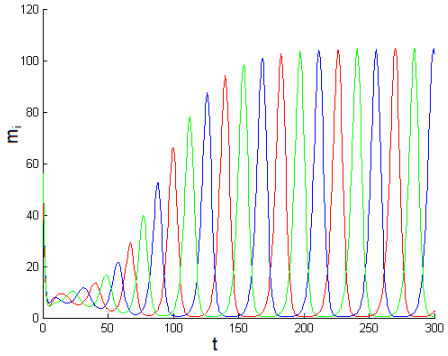
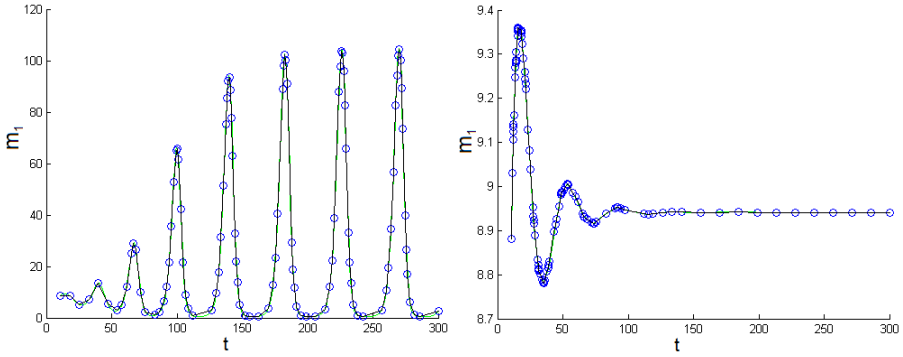


Figure 10: A typical signal produced by the system (4). Only variables m_1, m_2 and m_3 are depicted.



(a) $\alpha = 400, \beta = 0.2, \alpha_0 = 0.2, n = 2$ (b) $\alpha = 400, \beta = 0.2, \alpha_0 = 2, n = 2$

Figure 11: Sample runs of the system (4) for different sets of parameters. The initial values were $m_1 = 0.1, m_2 = 0.3, m_3 = 0.2, p_1 = 0.2, p_2 = 0.1, p_3 = 0.3$. Only the values of m_1 are depicted.

Formula (5) is satisfied by both runs in Figure 11. To avoid the case of damped oscillations, we can add the formula:

$$\psi \stackrel{\text{df}}{=} \mathbf{G}_{[10,200]} * \big[\big(\mathbf{F}_{[1,50]}\, m_i^* \leq m_i \big) \big]. \tag{6}$$

It ensures that the values reached in each period are not decreasing. By connecting formulae (5) and (6) we get the formula $\varphi \wedge \psi$. It does express the desired oscillatory behaviour in Figure 11a and it is not satisfied by signals of the type depicted in Figure 11b.

Another property of the repressilator can be expressed by the formula:

$$\mathbf{G}_{[0,270]} * \big[\big(\mathbf{F}_{[0,30]}\, (m_1^* + 1 > m_3 \wedge m_1^* - 1 < m_3) \big) \big], \tag{7}$$

which means that *"the values of m_1 precede the values of m_3 in such sense that for every value of m_1, m_3 reaches similar value in short time after m_1 did"* (Figure 12).

Full estimation of parameters could not be performed due to high time demands of the implemented monitoring algorithm [10]. A single run of the monitoring algorithm for the formulae (5), (6) or (7) over a signal sampled by 80 points took several hours on a regular PC. Reduction of the number of points describing the signal would lead to excessive loss of information. We have found out that the

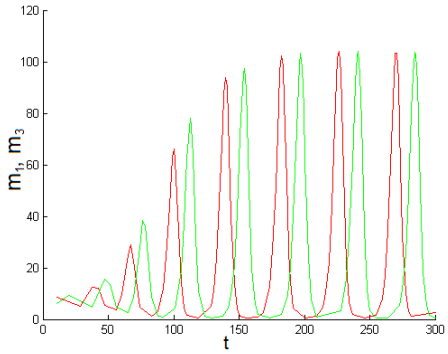


Figure 12: Sample run of the system (4) satisfying the formula (7). The parameters were $\alpha = 400, \beta = 0.2, \alpha_0 = 0.2, n = 2$ and the initial values $m_1 = 0.1, m_2 = 0.3, m_3 = 0.2, p_1 = 0.2, p_2 = 0.1, p_3 = 0.3$. Depicted variables are m_1 – red and m_3 – green.

bottleneck in efficiency of the prototype implementation lies in polyhedral operations performed by MPT. Since we work in 2D and need only a small subset of operations, we believe that our algorithm can be significantly accelerated when an optimal implementation is employed. Here we focused on demonstrating the applicability while leaving efficiency for future work.

Different task would be to ensure that the behaviour of concentrations of the fluorescent protein in the bacteria (Figure 9) is in correspondence with the model. While it could be done only manually in [14], if we had the data, we would be able to test the properties of the measurements identically as the properties of the signals produced by ODEs.

6 Conclusion

We have proposed an extended Signal Temporal Logic STL* motivated by the need to express properties of biological dynamical systems in a detail sufficient to distinguish different shapes of oscillation. We have provided a monitoring algorithm that approximately computes the truth value of an STL* formula for a given continuous (piece-wise linear) signal. The method has been prototyped in Matlab and the results achieved on a case study of oscillatory behaviour of the repressilator showed that the method satisfactorily works for signals generated by numerical simulation. However, the employed library MPT for polyhedral operations appears to be not efficient enough to satisfy the needs of practical usage.

For future work on the practical side, we plan to implement more efficient algorithms for the specific polyhedral operations we use in the monitoring procedure. Another straightforward direction of future development is lifting of the robustness measure [11] to the extended logic.

A very inspiring work is [12] where Time-Frequency Logic (TFL) is defined by shifting STL semantics to frequency domain. TFL provides another way to express permanent oscillations. However, non-pure oscillatory behaviour such as damped oscillations require specific elaboration in the time-domain. Joining TFL semantics with the STL* concept of real-value freezing can be an interesting step further.

References

[1] Accelera Organization, Inc. (2004): *SystemVerilog 3.1a Language Reference Manual*.

[2] R. Alur, T. Feder & T. A. Henzinger (1996): *The benefits of relaxing punctuality*. *J ACM* 43(1), pp. 116–146, doi:10.1145/227595.227602.

[3] R. Alur & T. A. Henzinger (1994): *A really temporal logic*. *J ACM* 41(1), pp. 181–203, doi:10.1145/174644.174651.

[4] P. Ballarini & M. L. Guerriero (2010): *Query-based verification of qualitative trends and oscillations in biochemical systems*. Theor. Comput. Sci. 411(20), pp. 2019–2036, doi:10.1016/j.tcs.2010.02.010.

[5] J. Barnat, L. Brim & D. Šafránek (2010): *High-performance analysis of biological systems dynamics with the DiVin E model checker*. Brief. in Bioinformatics 11, pp. 301–312, doi:10.1093/bib/bbp074.

[6] E. Bartocci, F. Corradini, E. Merelli & L. Tesei (2010): *Detecting synchronisation of biological oscillators by model checking*. Theoretical Computer Science 411(20), pp. 1999 – 2018, doi:10.1016/j.tcs.2009.12.019.

[7] G. Batt, D. Ropers, H. de Jong, J. Geiselmann, R. Mateescu, M. Page & D. Schneider (2005): *Validation of qualitative models of genetic regulatory networks by model checking: analysis of the nutritional stress response in Escherichia coli*. In: ISMB (Supplement of Bioinformatics), pp. 19–28.

[8] M. de Berg, O. Cheong, M. van Kreveld & M. Overmars (2008): *Computational Geometry: Algorithms and Applications*, 3 edition. Springer, Berlin.

[9] L. Calzone, N. Chabrier-rivier, F. Fages & S. Soliman (2006): *Machine learning biochemical networks from temporal logic properties*. Trans Comput Syst Biol 4220, pp. 68–94, doi:10.1007/11880646_4.

[10] P. Dluhoš (2012): *Specification and monitoring of oscillation properties in dynamical systems*. Master's thesis, Masaryk University. Available at `http://is.muni.cz/th/269281/fi_m/thesis.pdf`.

[11] A. Donzé & O. Maler (2010): *Robust satisfaction of temporal logic over real-valued signals*. In: *FORMATS'10*, Springer-Verlag, Berlin, Heidelberg, pp. 92–106, doi:10.1007/978-3-642-15297-9_9.

[12] A. Donzé, O. Maler, E. Bartocci, D. Nickovic, R. Grosu & S. Smolka (2012): *On Temporal Logic and Signal Processing*. Accepted to ATVA 2012.

[13] C. Eisner & D. Fisman (2008): *Augmenting a regular expression-based temporal logic with local variables*. In: *FMCAD '08*, IEEE Press, pp. 23:1–23:8, doi:10.1.1.140.3930.

[14] M. B. Elowitz & S. Leibler (2000): *A synthetic oscillatory network of transcriptional regulators*. Nature 403(6767), pp. 335–338, doi:10.1038/35002125.

[15] B. N. Kholodenko (2000): *Negative feedback and ultrasensitivity can bring about oscillations in the mitogen-activated protein kinase cascades*. Eur J Biochem 267(6), pp. 1583–1588, doi:10.1046/j.1432-1327.2000.01197.x.

[16] M. Kvasnica, P. Grieder & M. Baotić (2004): *Multi-Parametric Toolbox (MPT)*.

[17] O. Maler & D. Nickovic (2004): *Monitoring temporal properties of continuous signals*. In: Proc. of FORMATS-FTRTFT, Springer, pp. 152–166, doi:10.1.1.102.2905.

[18] O. Maler, D. Nickovic & A. Pnueli (2008): *Checking Temporal Properties of Discrete, Timed and Continuous Behaviors*. In: Pillars of Computer Science, LNCS 4800, Springer, pp. 475–505, doi:10.1007/978-3-540-78127-1_26.

[19] R. Mateescu, P. T. Monteiro, E. Dumas & H. de Jong (2011): *CTRL: Extension of CTL with regular expressions and fairness operators to verify genetic regulatory networks*. Theor. Comput. Sci. 412(26), pp. 2854–2883, doi:10.1016/j.tcs.2010.05.009.

[20] F. Miyoshi, Y. Nakayama, K. Kaizu, H. Iwasaki & M. Tomita (2007): *A Mathematical Model for the Kai-Protein-Based Chemical Oscillator and Clock Gene Expression Rhythms in Cyanobacteria*. Journal of Biological Rhythms 22(1), pp. 69–80, doi:10.1177/0748730406295749.

[21] D. Nickovic & O. Maler (2007): *AMT: a property-based monitoring tool for analog systems*. In: Proceedings of the 5th international conference on Formal modeling and analysis of timed systems, FORMATS'07, Springer-Verlag, Berlin, Heidelberg, pp. 304–319.

[22] A. Rizk, G. Batt, F. Fages & S. Soliman (2008): *On a Continuous Degree of Satisfaction of Temporal Logic Formulae with Applications to Systems Biology*. In: Proc. of CMSB'08, Springer, pp. 251–268, doi:10.1007/978-3-540-88562-7_19.

[23] J. Červený & L. Nedbal (2009): *Metabolic Rhythms of the Cyanobacterium Cyanothece sp. ATCC 51142 Correlate with Modeled Dynamics of Circadian Clock*. J Biol Rhythms 24(4), pp. 295–303, doi:10.1177/0748730409338367.