Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science 229 (2009) 41–58
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Model Checking Biological Oscillators

Ezio Bartocci,[1] Flavio Corradini,[2] Emanuela Merelli,[3] Luca Tesei[4]

Dipartimento di Matematica e Informatica
University of Camerino
Via Madonna delle Carceri, 9, Camerino (MC), 62032, Italy

Abstract

We define a subclass of timed automata, called oscillator timed automata, suitable to model biological oscillators. The semantics of their interactions, parametric w.r.t. a model of synchronization, is introduced. We apply it to the Kuramoto model. Then, we introduce a logic, Kuramoto Synchronization Logic (KSL), and a model checking algorithm in order to verify collective synchronization properties of a population of coupled oscillators.

Keywords: Spontaneous synchronization, Kuramoto model, oscillator timed automata, KSL, model checking

1 Introduction

Synchronization phenomena in large populations of interacting components are widely represented in nature and intensively studied as physical, biological, chemical, and social systems. In Biology, examples include networks of pacemaker cells in heart [15], nervous system [8], group of synchronously flashing fireflies [6], just to mention some of those analyzed by Strogatz in his exciting book [16]. Understanding a synchronized collective behavior is essential in Systems Biology especially for developing methods to control the dynamics of systems with desired properties [10].

The distributed synchronization of biological systems is commonly modeled using the theory of coupled oscillators proposed by several authors Art Winfree [18], Charles S. Peskin [15] and Yoshiki Kuramoto [11]. In this theory, each member of the population is modeled as a phase oscillator running independently at its own

[1] Email: ezio.bartocci@unicam.it
[2] Email: flavio.corradini@unicam.it
[3] Email: emanuela.merelli@unicam.it
[4] Email: luca.tesei@unicam.it

doi:10.1016/j.entcs.2009.02.004

frequency. The synchronization could be achieved coupling each oscillator to all the others and making them to interact with a certain strength. Whereas, the control is achieved either by introducing artificial oscillators or a new impulse, or by changing the parameters of individual oscillators. This approach gives rise to an artificial control strategy as it has been proposed by Wang in his recent work [17]. In the case of fireflies, an oscillator represents the internal clock dictating when to flash. Upon reception of a pulse from other oscillators, the clock is regulated. Over time, the synchronized behavior emerges, i.e. pulses of different oscillators are transmitted simultaneously, and then all fireflies simultaneously flash. The most successful attempt to model distributed synchronization has been proposed by Yoshiki Kuramoto. The Kuramoto's model, based on Winfree's ideas that mutual synchronization is a cooperative phenomenon, a temporal analogue of phase transition encountered in statistical physics, is beautiful and analytically tractable model. A wide description of the Kuramoto model can be found in [1].

In this paper, we define a subclass of timed automata, called *oscillator timed automata*, suitable to model oscillators. Then, we introduce an interaction semantics which is parametric w.r.t. a model of synchronization. Subsequently, we instantiate it to the Kuramoto model. The main advantage of our approach is that although the timed automata show different behaviors, w.r.t. their original ones, when interacting, we do not need to change their structure. This is obtained by using a global time measure, which is the one of the observer, and several internal relative time measures, one for each oscillator, which are then re-scaled to the global time.

We also provide a Kuromoto Synchronization Logic (KSL) in order to specify properties on synchronization, during a simulation, of Kuramoto based oscillators. Moreover, we propose a model checking algorithm for the logic. We test it by verifying some properties with a prototype simulator and model checker that we have implemented.

The paper proceeds as follows: Section 2 introduces related works and the Kuramoto model. Section 3 recalls timed automata, defines oscillator timed automata, and specifies the interaction semantics. Section 4 introduces the Kuramoto Synchronization Logic (KSL) and its model checking algorithm. Section 5 shows an example of analysis and Section 6 concludes outlining some directions for future work.

2 Background

2.1 Related work

During these last decades, several mathematical models have been proposed to study the spontaneous synchronization phenomena in a population of biological coupled oscillators [16]. These models have been inspired by real biological systems, ranging from the mutual synchronization of cardiac and circadian pacemaker cells to the rhythmically flashing of fireflies and wave propagation in heart, brain, intestine and nervous system. In these systems, mutual synchronization could be performed both through episodic impulses and through smooth interactions.

The first case, in which a population of the so called pulse-coupled oscillators communicate by sudden pulse-like interactions - i.e. a neuron that fires - was first studied by Peskin [15], who proposed a model of the mutual synchronization of sinotrial node pacemaker cells. He worked with identical oscillators and he conjectured that for any arbitrary conditions, they would all end up firing in unison. He proved this property for $N = 2$ oscillators and later Mirollo and Strogatz [14] demonstrated that the conjecture holds for all N. Peskin also conjectured that synchronization would occur even if the oscillators were not quite identical, but that problem remains still open.

For the second case, in which the interactions between oscillators are smooth, a first approach was proposed by Winfree [18] that introduced a model of nearly identical, weakly coupled limit-cycle oscillators. Using numerical simulation, he discovered that for these starting conditions, the system behaves incoherently, with each oscillator running at its natural frequency. He also found that, as the coupling is increased, the unsynchronized incoherence continues until a certain threshold, when a group of oscillators jump suddenly into synchrony.

Starting from Winfree's results and assumptions, Kuramoto began working with collective synchronization phenomena and he proposed a refined model [11,12] providing some analytical tools in order to render the problem more tractable and the synchronization measurable.

2.2 Kuramoto model

The Kuramoto model of synchronization [13] describes the dynamics of a set of N interacting phase oscillators θ_i with natural frequencies ω_i and initial phases θ_i^0.

The standalone evolution of the i-th oscillator is described by $\theta_i(t) = \omega_i t + \theta_i^0$. Intuitively each oscillator i can be visualized as a point moving on a circle of radius 1 with angular speed ω_i starting at angle θ_i^0.

When the oscillators interact they tend to adapt themselves, by accelerating or decelerating, with respect to the phases and the frequencies of the others. This can be viewed as a process of collective synchronization which ends, under certain conditions, in a total synchronous behavior. In the metaphor of the points moving on the circle, when the system becomes synchronized the points move around in sync, meaning that the phase differences remain constant. If the oscillators have identical natural frequencies, they are also null.

By Kuramoto, the evolution of the interacting i-th oscillator is given by the following equation:

$$\dot{\theta}_i = \omega_i + \frac{K}{N}\sum_{j=1}^{N}\sin(\theta_j - \theta_i), \quad i = 1, \cdots, N \tag{1}$$

where K is the coupling strength depending on the type of interaction. A primary basic condition to the possibility of synchronization is that the natural frequencies of the N oscillators are equal or chosen from a lorentzian probability density given by:

$$g(\omega) = \frac{\gamma}{\pi[\gamma^2 + (\omega - \omega_0)^2]}$$

where γ is the width and ω_0 is the median.

In his analysis [13], Kuramoto provided a measure of synchronization by defining the complex order parameters r and ψ as:

$$re^{i\psi} = \frac{1}{N}\sum_{j=1}^{N} e^{i\theta_j} \tag{2}$$

where r is the magnitude of the centroid of the points and ψ indicates the average phase. The radius r represents the phase-coherence of the population of the oscillators and it is a convenient measure of the extent of synchronization in the limit $N \to \infty$ and $t \to \infty$. If all oscillators are in sync, $r = 1$ when all the ω_i's are the same while $r \approx 1$ when the natural frequencies are not identical. On the other hand, when all oscillators are completely out of phase with respect to each other the value of r remains close to 0 most of the time.

In particular Kuramoto found that:

$$r = \begin{cases} 0 & K < K_c \\ \sqrt{1 - (K_c/K)} & K \geq K_c \end{cases}$$

where $K_c = 2\gamma$. This means that the oscillators remain completely desynchronized until the coupling strength K exceeds a critical value K_c. After that, the population splits into a partially synchronized state consisting of two groups of oscillators: a synchronized group that contributes to the order parameter r, and a desynchronized group whose natural frequencies lie in the tails of the distribution $g(\omega)$ and are too extreme to be entrained. With further increases in K, more and more oscillators are recruited into the synchronized group, and r grows accordingly.

Usually, to approximate Equation (1) a suitable discretization is made by fixing a small time interval dt to perform a numerical simulation. See Section 4 for a discussion on the numerical methods used in our approach.

3 Automata model

In this section we show how oscillators can be modeled by timed automata and how their interaction semantics can be defined, basing on a synchronization model, without changing the structure of the standalone automata.

3.1 *Timed automata*

Timed Automata [2] are an established formalism for modeling and verifying real-time systems. They allow strict quantitative real-time constraints to be expressed. This characteristic will be used to model oscillators.

In this section we introduce the basic machinery on timed automata that we need for our purposes. The idea of clock variables is central in the framework of timed automata. A *clock* is a variable that takes values from the set $\mathbb{R}^{\geq 0}$. The clocks measure time as it elapses. All the clocks of a given system advance at the same rate: when increasing, they can be viewed as functions on time whose derivative is equal to 1. Clock variables are ranged over by $x, y, z, \ldots$ and we use $\mathcal{X}, \mathcal{X}', \ldots$ to denote sets of clocks. A *clock valuation* over $\mathcal{X}$ is a function assigning a positive natural number to every clock. The set of valuations of $\mathcal{X}$, denoted by $\mathcal{V}_{\mathcal{X}}$, is the set of total functions from $\mathcal{X}$ to $\mathbb{R}^{\geq 0}$. Clock valuations are ranged over by $\nu, \nu', \ldots$. Given $\nu \in \mathcal{V}_{\mathcal{X}}$ and $\delta \in \mathbb{R}^{>0}$, we use $\nu + \delta$ to denote the valuation that maps each clock $x \in \mathcal{X}$ into $\nu(x) + \delta$.

Clock variables can be reset during the evolution of the system when certain actions are performed or certain events occur. The reset consists in instantaneously set the value of a clock to 0. Immediately after this operation the clock restarts to measure time at the same rate as the others. The reset is useful to measure the time elapsed since the action/event that reset the clock occurred. Given a set $\mathcal{X}$ of clocks, a *reset* γ is a subset of $\mathcal{X}$. The set of all resets of clocks in $\mathcal{X}$ is denoted by $\Gamma_{\mathcal{X}}$ and reset sets are ranged over by $\gamma, \gamma', \ldots$ Given a valuation $\nu \in \mathcal{V}_{\mathcal{X}}$ and a reset γ, we let $\nu \backslash \gamma$ be the valuation that assign the value 0 to every clock in γ and assign $\nu(x)$ to every clock $x \in \mathcal{X} - \gamma$.

The timed behavior of the system is expressed using constraints associated to the edges of the automaton. Such constraints depend on the actual values of the clock variables of the system. Given a set $\mathcal{X}$ of clocks, the set $\Psi_{\mathcal{X}}$ of *clock constraints* over $\mathcal{X}$ are defined by the following grammar: $\psi \ ::= \ \mathit{true} \mid \mathit{false} \mid x \,\#c \mid x - y \,\#\, c \mid \psi \wedge \psi$ where $x, y \in \mathcal{X}$, $c \in \mathbb{N}$, and $\# \in \{<, >, \leq, \geq, =\}$. A satisfaction relation $\models$ is defined such that $\nu \models \psi$ if the values of the clocks in ν satisfy the constraint ψ in the natural interpretation.

Definition 3.1 A *Timed Automaton* T is a tuple $(Q, \Sigma, \mathcal{E}, q_0, \mathcal{X}, \mathit{Inv})$, where: Q is a finite set of locations, Σ is a finite alphabet of symbols, $\mathcal{E}$ is a finite set of edges, q_0 is the initial state, $\mathcal{X}$ is a finite set of clocks, and Inv is a function assigning to every $q \in Q$ an *invariant*, i.e. a clock constraint ψ such that for each clock valuation $\nu \in \mathcal{V}_{\mathcal{X}}$ and for each $\delta \in \mathbb{R}^{>0}$, $\nu + \delta \models \psi \Rightarrow \nu \models \psi$. Constraints having this property are called *past-closed*.

Each edge $e \in \mathcal{E}$ is a tuple in $Q \times \Psi_{\mathcal{X}} \times \Gamma_{\mathcal{X}} \times \Sigma \times Q$. If $e = (q, \psi, \gamma, a, q')$ is an edge, q is the *source*, q' is the *target*, ψ is the *constraint*, a is the *label*, and γ is the *reset*.

We use timed automata with invariants on the states, usually called timed safety automata [9], that are the most common in the modeling and verification tools. They incorporate a strong notion of fairness, due to the invariants, which will be useful for the definition of oscillator timed automata in Section 3.2 and of the interaction semantics in Section 3.3.

Figure 1(a) shows a sample timed automaton with three states $0, 1, 2$. The set of clocks is $\{x\}$, the alphabet is $\{a, b\}$, 0 is the initial state, and the invariant of state

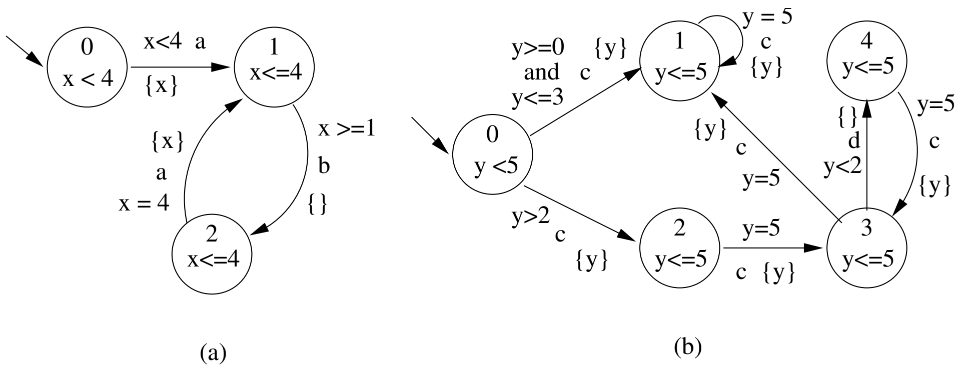

Fig. 1. Two oscillator timed automata

0 is $x < 4$. There is an edge from state 0 to state 1 with clock constraint $x < 4$, label a and reset set $\{x\}$.

The semantics of a timed automaton $T = (Q, \Sigma, \mathcal{E}, q_0, \mathcal{X}, Inv)$, is a labeled transition system $S(T)$ whose states - ranged over by $s, s', \ldots$ - are pairs (q, ν), where $q \in Q$ is a location of T, and $\nu \in \mathcal{V}_{\mathcal{X}}$ is a clock valuation. The transition relation is defined by the following rules:

$$\text{T1} \quad \frac{\delta \in \mathbb{R}^{>0} \qquad \nu + \delta \models Inv(q)}{(q, \nu) \xrightarrow{\delta} (q, \nu + \delta)} \qquad \text{T2} \quad \frac{(q, \psi, \gamma, a, q') \in \mathcal{E}, \nu \models \psi}{(q, \nu) \xrightarrow{a} (q', \nu \backslash \gamma)}$$

Rule T1 let δ time units to elapse, provided that the invariant of the current location will be satisfied at the reached state. We call the transitions performed using this rule *δ-transitions.* Rule T2 describe a transition, labelled by a, of the automaton which is possible only if the current clock evaluation ν satisfies the clock constraint of the edge. The effect of the transition is to go in the targe location q' where the clocks in the reset set γ have been assigned to 0. We call the transitions performed using this rule *a-transitions.*

The initial state of $S(T)$ is (q_0, ν_0) where ν_0 is the clock valuation assigning 0 to all clocks.

A prefix of a possible behavior of the automaton in Figure 1(a) is $r_{ex} = (0, [x = 0]) \xrightarrow{3.4} (0, [x = 3.4]) \xrightarrow{a} (1, [x = 0]) \xrightarrow{1.2} (1, [x = 1.2]) \xrightarrow{b} (2, [x = 1.2]) \ldots$

Let $T = (Q, \Sigma, \mathcal{E}, q_0, \mathcal{X}, Inv)$ be a timed automaton and let $r = s_0 \xrightarrow{l_0} s_1 \xrightarrow{l_1} \cdots$ be an *infinite* derivation of $S(T)$ where $s_0 = (q_0, \nu_0)$ is an initial state.

- The *time sequence* $t_0\, t_1\, t_2 \cdots$ of the times elapsed from state s_0 to every state $s_i = (q_i, \nu_i)$ in r is defined as follows:
  $t_{-1} = 0$
  $$t_{i+1} = t_i + \begin{cases} 0 \text{ if } l_i \in \Sigma \\ l_i \text{ otherwise} \end{cases}$$

- The *label sequence* of r is the sequence of the transitions occurred during r, including the elapsed times, from the initial state: $(l_0, t_0)(l_1, t_0)\cdots$
- The *action sequence* of r is the projection of the label sequence of r on the pairs $\{(l_i, t_i) \mid i \geq 0,\ l_i \in \Sigma\}$
- If $a \in \Sigma$ the *a-sequence* of r is the projection of the label sequence of r on the pairs $\{(l_i, t_i) \mid i \geq 0,\ l_i = a\}$

The time sequence of r_{ex} is $t_{-1} = 0, t_0 = 3.4, t_1 = 3.4, t_2 = 4.6, t_3 = 4.6, \ldots$ The label sequence is $(3.4, 3.4)(a, 3.4)(1.2, 4.6)(b, 4.6)\cdots$ The action sequence is $(a, 3.4)(b, 4.6)\cdots$ The b-sequence is $(b, 4.6)\cdots$

3.2 Oscillator timed automata

The following definition characterizes the subclass of timed automata that are suitable to represent oscillators.

Definition 3.2 A timed automaton T is called *oscillator timed automaton* on a *distinguished action* $a \in \Sigma$ with *period* $p \in \mathbb{R}^{>0}$ if and only if all the following conditions hold:

(i) for all $\vartheta \in \mathbb{R}^{\geq 0}$ such that $0 \leq \vartheta < p$ there exists a prefix of an infinite derivation r of $S(T)$ of the form $s_0 \xrightarrow{l_0} s_1 \xrightarrow{l_1} \cdots \xrightarrow{l_{n-1}} s_n \xrightarrow{l_n} s_{n+1}$ where all l_i $(i = 0, \ldots, n-1)$ are in $\mathbb{R}^{>0}$ (i.e. they are δ-transitions) **and** $l_n = a$

(ii) every infinite derivation r of $S(T)$ has a prefix of the form (i)

(iii) every infinite derivation r of $S(T)$ has an infinite a-sequence of the form $(a, \vartheta)(a, 1 \cdot p + \vartheta)(a, 2 \cdot p + \vartheta)(a, 3 \cdot p + \vartheta)\cdots$ for some delay ϑ

(iv) every finite derivation of $S(T)$ is a prefix of an infinite derivation r of $S(T)$

This definition constrains an oscillator timed automaton to have an initial state from which the oscillation can start with every delay (i), to always perform the distinguished action as the first action (ii), to regularly repeat the distinguished action every period from the first action on (iii), and to not have dying paths (iv), that is paths ending in a state where time can not proceed.

Such a behavior is a convenient representation of an oscillator with its own initial phase θ_0 $(0 \leq \theta_0 < 2\pi)$, which we represent here as a time delay ϑ instead of an angle, and its own frequency ω, which we represent here as a period p. They are related as follows: $\omega = \frac{2\pi}{p}$ and $\theta_0 = \frac{2\pi}{p}\vartheta$.

Figure 1(a) is an oscillator timed automaton on the distinguished action a with period 4. In Figure 1(b) we show a slightly more complex automaton which is an oscillator timed automaton on the distinguished action c with period 5. Note that there is non-determinism on the choice of the initial non-δ-transition. Moreover, there may be both an infinite self-loop on state 1 and a cycle between states 3 and 4 that eventually could end in the loop of state 1. In general, oscillator timed automata can be very complex automata performing several actions and involving cycles other than the one we focus on. The choice of the distinguished action identifies the particular observation with which an external observer recognizes the oscillating behavior.

3.3 Interacting oscillator timed automata

We want to describe the interaction among several oscillator timed automata. The definition we give here is parametric with respect to the model of synchronization. After the parametric definition we introduce the particular instance that uses the Kuramoto model.

Let $T_1, \ldots, T_N$ be oscillator timed automata on distinguished actions $a_1, \ldots, a_N$ with periods $p_1, \ldots, p_N$. We suppose we are given, randomly determined or specifically chosen, N initial delays $\vartheta_1, \ldots, \vartheta_N$.

At every instant of the interaction process we need to keep track of the current position of each oscillator in its standalone cycle. To do this we define a simple transformation: we add a new clock $\overline{x}_i$ to every T_i and, to guarantee a correct measure, we modify each T_i in such a way that $\overline{x}_i$ is reset whenever the distinguished action a_i is performed by T_i. This can be easily done by replacing each edge $(q, \psi, \gamma, a_i, q')$ of T_i by $(q, \psi, \gamma \cup \{\overline{x}_i\}, a_i, q')$.

The previous transformation ensures that, after a proper initialization, $\overline{x}_i$ measures, at every point of the evolution of $S(T_i)$, the time elapsed since the last occurrence of a_i. By the assumption that T_i is an oscillator timed automaton, we can also state that p_i minus the value of $\overline{x}_i$ measures the remaining time to the next occurrence of a_i. The initialization process is described below.

3.3.1 Steps of activity

In order to describe the interaction semantics we need to define the *steps of activity* of a single automaton.

Let $T = (Q, \Sigma, \mathcal{E}, q_0, \mathcal{X}, Inv)$ be a timed automaton. Given $\Delta \in \mathbb{R}^{>0}$ we define a transition relation $\xrightarrow[\Delta]{\ell}$ between two states of $S(T)$. We say that $(q, \nu) \xrightarrow[\Delta]{\ell} (q', \nu')$ if and only if there exists a *finite* derivation d of $S(T)$ of the form $(q, \nu) = (q^0, \nu^0) \xrightarrow{l_0} (q^1, \nu^1) \xrightarrow{l_0} \cdots \xrightarrow{l_{n-1}} (q^n, \nu^n) = (q', \nu')$ such that:

- ℓ is the string $l_0 l_1 \cdots l_{n-1}$
- Δ is the sum of the times elapsed during d, i.e. $\Delta = \sum_{i=0, l_i \in \mathbb{R}^{>0}}^{n-1} l_i$

The *time sequence* of d is defined as the time sequence of r in Section 3.1, starting from state (q, ν) and ending on state (q', ν'). The *label sequence*, *action sequence* and *a-sequence* of d are defined in the same way. Note that in this case the sequences are all finite. Moreover, given a step $(q, \nu) \xrightarrow[\Delta]{\ell} (q', \nu')$, we define a function $\mathcal{A}$ such that $\mathcal{A}(\ell)$ gives the action sequence of the derivation d associated to the step.

The steps of activity allow us to group a sequence of transitions of $S(T)$ into a single transition $\xrightarrow[\Delta]{\ell}$. We can decide the amount of time (Δ) of our observation (ℓ) of the automaton behavior. Note that a step of activity is always possible only if the automaton has not dying paths. Since we use this transition relation only with oscillator timed automata, this is guaranteed by Definition 3.2:

Proposition 3.3 *Let T be an oscillator timed automaton and let $(q_0, \nu_0) \xrightarrow{l_0} \cdots \xrightarrow{l_n} (q, \nu)$ be a finite derivation of $S(T)$. Then, given $\Delta \in \mathbb{R}^{>0}$, it is always possible to perform a step of activity $(q, \nu) \xrightarrow[\Delta]{\ell} (q', \nu')$.*

Proof. By property (iv) of oscillator timed automata expressed in Definition 3.2.□

3.3.2 Initialization

The initialization step of the interaction must ensure that we start each transition system $S(T_i)$ in a state with complete information about the delay and, thus, about the values of $\overline{x}_i$ and the other clocks. To achieve this requirement we need to perform a preliminary collective derivation of all $S(T_i)$. Before describing the procedure we need to introduce the following proposition.

Proposition 3.4 *Let $T = (Q, \Sigma, \mathcal{E}, q_0, \mathcal{X}, Inv)$ be an oscillator timed automaton on the distinguished action $\overline{a}$ with period p. For all delays $0 \leq \vartheta < p$, there exists an edge $(q_0, \psi, \gamma, \overline{a}, q') \in \mathcal{E}$ and a prefix $d = s_0 \xrightarrow{l_0} s_1 \xrightarrow{l_1} \cdots \xrightarrow{l_{n-1}} s_n \xrightarrow{l_n} s_{n+1}$ of an infinite derivation r of $S(T)$ such that:*

- *all l_i, $i = 0, \ldots, n-1$, are in $\mathbb{R}^{>0}$, $\sum_{i=1}^{n-1} l_i = \vartheta$ and $l_n = \overline{a}$*
- *$s_n = (q_0, \nu_0 + \vartheta)$ and $s_{n+1} = (q', (\nu_0 + \vartheta)/\gamma)$*

Proof. The properties follow easily by Definition 3.2 and by the semantics of a timed automaton. □

The previous Proposition implies that we can effectively determine a state s_{n+1} of $S(T)$ from which the actual interaction of the automaton with the others can start.

Let $B_1, B_2, \ldots, B_K$, $K \leq N$, be a sequence of non-empty sets partitioning $\{1, \ldots, N\}$ such that for all k, $k = 1, 2, \ldots, K-1$, if $i \in B_k$ and $j \in B_{k+1}$ then $\vartheta_i < \vartheta_j$; i.e. every B_k contains the indexes of the automata with equal values of initial delay and the sequence arranges the indexes of automata in such a way that their initial delays are in strict ascending order.

The preliminary collective derivation is as follows. It is straightforward, but requires a lot of details: for the sake of brevity we only give a brief informal description.

We consider the initial states $s_0^1, s_0^2, \ldots, s_0^N$ of each $S(T_i)$, where every T_i has been transformed inserting the new clock and modifying the edges as specified above.

The process is a sequence of K steps. At the first step every $S(T_i)$ performs zero or more δ-transitions in order to reach a state of the form $(q_0, \nu_0 + \vartheta)$ where ϑ is the delay of the first group of oscillators whose indexes are in B_1. Then, the automata in this group perform a further single a-transition (where a is the distinguished action) to reach a state of the form $(q', (\nu_0 + \vartheta)/\gamma)$, as guaranteed by Proposition 3.4. From now on every automaton of this group is considered "activated" and continues with a sequence of transitions of the form $\xrightarrow[\Delta]{\ell}$ where Δ is given by the times specified in the subsequent steps.

At the i-th step the time that has to pass is $\vartheta = \vartheta_k - \vartheta_j$, where $j \in B_{i-1}$ and $k \in B_i$. The previously "activated" automata perform a transition $\xrightarrow[\vartheta]{\ell}$, while all the others perform δ-transitions for a total of ϑ time units. Then, the automata whose indexes are in B_i becomes "activated" performing their first a-transition (a distinguished). Again, this possibility is guaranteed by Proposition 3.4.

At the end of this process we group the N reached states into a tuple $\langle s_0^1, s_0^2, \ldots, s_0^N\rangle$ which is the initial configuration of the interaction semantics. Note that, during the process, the early "activated" automata could have performed other a-transitions, even distinguished ones. This does not influence the subsequent interactions, because there have not been perturbations due to the synchronization function. The important fact to remark is that the states of the starting tuple all have pertinent values for the clocks $\overline{x}_i$ and represent running oscillators with the chosen delays and periods.

3.3.3 Interaction semantics

Now we can introduce the behavior of N oscillator timed automata running in parallel and interacting using a generic discretized model, which is represented by an interaction function $\mathcal{I}$.

The main advantage of the interaction semantics is that it maintains the structure of the automata T_i's, though it defines a perturbed behavior of them due to the interaction. This is achieved by using a *global* time, which is the time measure of the observer, together with N different *relative* times, which are the internal time measures of each oscillator timed automata involved. This means that if, according to the interaction, T_i has to decelerate and T_j has to accelerate during the current slice of global time dt, then T_i performs a step of activity whose duration is shorter than dt, while T_j performs a longer one. The acceleration and deceleration is determined by the interaction function $\mathcal{I}$ of the particular chosen model of synchronization. The global observer detects perturbed behaviors because the timestamps of the actions performed by each interacting T_i are re-scaled to the magnitude of the global time slice dt.

Now we can introduce a transition relation, which we call *progress relation*, between configurations, i.e. tuples $\langle s_1, s_2, \ldots, s_N\rangle$ where each s_i is a state of $S(T_i)$. The rule defining the relation is the following:

$$\frac{\forall i = 1, 2, \ldots N \quad \Delta_i = \mathcal{I}(i, dt, s_1, s_2, \ldots, s_N) \quad s_i \xrightarrow[\Delta_i]{\ell_i} s_i' \quad \lambda^i = \mathcal{SC}(\mathcal{A}(\ell_i), \Delta_i, dt)}{\langle s_1, s_2, \ldots, s_N\rangle \xrightarrow[dt]{\lambda^1, \lambda^2, \ldots, \lambda^N} \langle s_1', s_2', \ldots, s_N'\rangle}$$

where the re-scaling function $\mathcal{SC}$ is defined as follows:

$$\mathcal{SC}((a_0, t_0)(a_1, t_1)\cdots(a_k, t_k), \Delta, dt) = (a_0, \frac{t_0}{\Delta}\cdot dt)(a_1, \frac{t_1}{\Delta}\cdot dt)\cdots(a_k, \frac{t_k}{\Delta}\cdot dt)$$

The behavior of the interacting oscillator timed automata $T_1, T_2, \ldots, T_N$ is then defined as an infinite derivation:

$$\rho = \langle s_0^1, s_0^2, \ldots, s_0^N \rangle \xrightarrow[dt]{\lambda_0^1, \lambda_0^2, \ldots, \lambda_0^N} \langle s_1^1, s_1^2, \ldots, s_1^N \rangle \xrightarrow[dt]{\lambda_1^1, \lambda_1^2, \ldots, \lambda_1^N} \cdots$$

To construct the trace associated to ρ, we need a function Ω that merges the scaled action sequences $\lambda^1, \lambda^2, \ldots, \lambda^N$ of every step of progress into one action sequence in which the timestamps of each action are in ascending order. Moreover we need a function $\mathcal{T}(\tau, (a_0, t_0)(a_1, t_1) \cdots (a_k, t_k)) = (a_0, t_0+\tau)(a_1, t_1+\tau) \cdots (a_k, t_k+\tau)$ that adds a given amount of time τ to the timestamps of an action sequence.

Definition 3.5 Given N interacting oscillator timed automata $T_1, \ldots, T_N$, the *trace* associated to a derivation $\rho = \langle s_0^1, s_0^2, \ldots, s_0^N \rangle \xrightarrow[dt]{\lambda_0^1, \lambda_0^2, \ldots, \lambda_0^N} \langle s_1^1, s_1^2, \ldots, s_1^N \rangle \xrightarrow[dt]{\lambda_1^1, \lambda_1^2, \ldots, \lambda_1^N} \langle s_2^1, s_2^2, \ldots, s_2^N \rangle \xrightarrow[dt]{\lambda_2^1, \lambda_2^2, \ldots, \lambda_2^N} \cdots$ is the following action sequence:

$$\Omega(\lambda_0^1, \lambda_0^2, \ldots, \lambda_0^N)\ \mathcal{T}(1 \cdot dt, \Omega(\lambda_1^1, \lambda_1^2, \ldots, \lambda_1^N))\ \mathcal{T}(2 \cdot dt, \Omega(\lambda_2^1, \lambda_2^2, \ldots, \lambda_2^N))\ \cdots$$

The *observable behaviors* of the interacting oscillator timed automata are all possible traces.

Note that traces contain all the actions performed by all the interacting automata. Moreover, the timestamp of each action is the one perceived by the global observer, which can therefore detect the perturbations, due to interaction, on the original standalone behaviors. Projecting a trace on distinguished actions only allows to observe the synchronization process if, eventually, all the distinguished actions occur at the same times.

3.3.4 Kuramoto interaction

In this section we instantiate the interaction semantics given above to the discretized Kuramoto model of synchronization introduced in Section 2.2.

In the automata model we use periods instead of natural frequencies and initial delays instead of initial phases. Since the Kuramoto model is defined in terms of angular frequency and angular acceleration, we need to do a simple transformation to express the acceleration $\frac{K}{N} \sum_{j=1}^{N} \sin(\theta_j - \theta_i)$ of the i-th oscillator in terms of shorter or longer duration Δ_i of the step of activity of T_i at each progress of length dt. This is what the function $\mathcal{I}$ does, which is described in the following.

First, note that each configuration $\langle s_1, s_2, \ldots, s_N \rangle$ is a snapshot of the situation of each oscillator at a given point of time t. This situation can be depicted in a circle of radius 1 associating to each automaton T_i, modulo $2\pi h$ for some $h \in \mathbb{N}$, the angle $\theta_i(t) = \frac{u_i}{p_i} \cdot 2\pi$, where u_i is the time elapsed since the last occurrence of the distinguished action a_i. By the transformation described at the beginning of Section 3.3, we know that u_i is precisely the value of the clock $\overline{x}_i$ that can be derived from $s_i = (q, \nu)$ as $\nu(\overline{x}_i)$. Note that this is true also in the starting configuration (for which we consider $t = 0$), as we showed in Section 3.3.2.

Using this transformation we can calculate the quantity

$$\alpha_i = \frac{K}{N} \sum_{j=1}^{N} \sin(\theta_j(t) - \theta_i(t))$$

for each automaton T_i. Then, according to the discretization of the Kuramoto model (see Section 2.2), during the time slice dt the i-th oscillator has to move forward, in the circle representation, of an angle $(\omega_i + \alpha_i)dt$, where $\omega_i = \frac{2\pi}{p_i}$ is its natural frequency. The time Δ_i the automaton T_i has to consume to do this is such that $(\omega_i + \alpha_i)dt = \omega_i \Delta_i$. Thus, $\Delta_i = dt + \frac{\alpha_i dt}{\omega_i}$, which is the value of $\mathcal{I}(i, dt, s_1, s_2, \ldots, s_N)$.

4 Logic for biological oscillators

In this section, we introduce a logic, Kuramoto Synchronization Logic (KSL), in order to specify and detect collective synchronization properties in a population of oscillator timed automata. We provide the syntax, the semantics and a model checking algorithm for KSL. This logic has the same main temporal and logical operators as Linear Temporal Logic (LTL). However, while LTL use a finite state automaton, the Kripke structure, as a model, KSL use an uncountable state model. In KSL, atomic propositions are given in terms of equalities or inequalities about linear combinations of state variables. This form allows us to easily describe some useful properties about synchronization. We add a special operator $D.$, inspired by the freeze quantification of [3], to store state variables in a certain step of the simulation that could be compared with the state variables of a successive step. Moreover, we use a bounded version of the until operator U adding to it a constraint on the maximum time interval to be considered.

4.1 Discrete time Kuramoto model

Our system model is a discrete time version of the Kuramoto model (see Equation (1)) which can be represented by a tuple $\mathcal{M} = (\Omega, \Theta, \mathcal{D}, \mathcal{D}^{(*)}, K, r)$, where $\Omega = \{\omega_1, \cdots, \omega_n\}$ is the vector of *frequencies* of the oscillator timed automata, $\Theta = \{\theta_1, \cdots, \theta_n\}$ is the vector of the *initial phases*, $\mathcal{D} = \{\mathrm{d}_1, \cdots, \mathrm{d}_n\}$ is the vector of the *remaining times* to accomplish the distinguished actions, $\mathcal{D}^{(*)}$ are the set of vectors of the *stored remaining times* during simulation, $K \in \mathbb{R}$ is the *interaction constant* between the oscillators automata, and r is the *phase coherence* calculated at current time as in Equation (2). We overload the definition of θ_i, d_i and r with the functions $\theta_i : \mathbb{N} \to \mathbb{R}$, $d_i : \mathbb{N} \to \mathbb{R}$ and $r : \mathbb{N} \to \mathbb{R}$ that map the discrete time t to the value of synchronization phases, remaining times and phase coherence.

The relations among synchronization phases and remaining times are given by the following equations:

$$\theta_i(t + dt) = \theta_i(t) + (\omega_i + \sum_{j=1}^{n} \frac{K}{n} \sin(\theta_i(t) - \theta_j(t))dt \tag{3}$$

$$d_i(t+dt) = \frac{(2q\pi - \theta_i(t+dt))}{\omega_i} \text{ such that } q = min(z \in \mathbb{N} : 2z\pi \leq \theta_i(t+dt))$$

If we consider dt as the time unit in the discretization of time we can rewrite Equation ()3) as follows:

$$\theta_i(t) = \theta_i(0) + \sum_{l=0}^{t-1} \left(\omega_i + \sum_{j=1}^{n} \frac{K}{n} sin(\theta_i(l) - \theta_j(l)) \right) dt \tag{4}$$

where $\theta_i(0)$ is the *synchronization phase* at the beginning of the simulation.

Although Equation (3) and (4) are very intuitive to understand and simple to be calculated, they need a very small dt to provide a good approximation, increasing dramatically the number of simulation steps. Consequently, in the simulator we implemented, we used the Runge-Kutta $4th$ order method [7] to obtain the best compromise between the approximation and the speed of computation.

4.2 Syntax and semantics of Kuramoto Synchronization Logic

The syntax of the Kuramoto Synchronization Logic is as follows:

$$\begin{aligned} \phi &::= T \mid F \mid p \mid \neg\phi \mid \phi \wedge \phi \mid X\phi \mid \phi\, U_{\prec m}\, \phi \mid D^{(h)}.\phi \\ p &::= \sum c_i v_i \sim b \mid r \sim b \end{aligned}$$

where p is an atomic proposition, c_i and b are real numbers, $v_i \in \mathcal{D} \cup \mathcal{D}^{(*)}$ are time remaining variables, $\sim \in \{<, \leq, >, \geq, =\}$, and $\prec \in \{<, \leq\}$. A KSL formula has atomic propositions, logical connectives $\neg$ $\wedge$, temporal connectives X (next) $U_{\prec m}$ (bounded until), and a freeze connective D. As usual, we introduce shorthands by defining the following derivative logical and temporal operators:

$\psi \vee \phi \Leftrightarrow \neg(\neg\psi \wedge \neg\phi)$	or ($\vee$)
$\psi \rightarrow \phi \Leftrightarrow \neg\psi \vee \phi$	implies ($\rightarrow$)
$\psi \leftrightarrow \phi \Leftrightarrow (\psi \rightarrow \phi) \wedge (\phi \rightarrow \psi)$	equivalent ($\leftrightarrow$)
$\Box_{\prec m}\psi \Leftrightarrow \neg(F\ U_{\prec m}\neg\psi)$	bounded always ($\Box$)
$\Diamond_{\prec m}\psi \Leftrightarrow T\ U_{\prec m}\psi$	bounded eventually ($\Diamond$)

Figure 2 shows the semantics of the basic operators. Concerning the temporal operators, $X\phi$ is true if and only if ϕ is true at next simulation step, and $\psi\, U_{\prec m}\, \phi$ is true if and only if ψ is continuously true - in subsequent steps - until ϕ becomes true, which must happen within m (if $\prec$ is $\leq$) or $m-1$ (if $\prec$ is $<$) steps.

Actually, the until operator of KSL can be expressed using the next operator. This is because the operator is always bounded.

$$
\begin{array}{lcl}
\mathcal{M}, t \models T & \Leftrightarrow & \mathcal{M}, t \not\models F \\
\mathcal{M}, t \models \sum_i c_i \cdot v_i \sim b & \Leftrightarrow & \sum_i c_i \cdot \mathcal{M}(v_i) \sim b \\
\mathcal{M}, t \models r \sim b & \Leftrightarrow & \mathcal{M}(r) \sim b \\
\mathcal{M}, t \models \neg\phi & \Leftrightarrow & \mathcal{M}, t \not\models \phi \\
\mathcal{M}, t \models \psi \wedge \phi & \Leftrightarrow & \mathcal{M}, t \models \psi \text{ and } \mathcal{M}, t \models \phi \\
\mathcal{M}, t \models X\ \phi & \Leftrightarrow & \mathcal{M}, t+1 \models \phi \\
\mathcal{M}, t \models \psi\ U_{\prec m}\ \phi & \Leftrightarrow & \exists s_2 : 0 \leq s_2 \prec m \text{ such that } \mathcal{M}, t+s_2 \models \phi \text{ and} \\
 & & \mathcal{M}, t+s_1 \models \psi \text{ for } s_1 = 0, \cdots, s_2 - 1 \\
\mathcal{M}, t \models D^{(h)}.\phi & \Leftrightarrow & \mathcal{M}_{[\mathcal{D}^{(*)} := \mathcal{D}^{(*)} \cup \{d_1^h(t), \cdots, d_n^h(t)\}]}, t \models \phi
\end{array}
$$

Fig. 2. Semantics of Kuramoto Synchronization Logic

Proposition 4.1 *Let $\psi\ U_{\prec m}\ \phi$ be a formula of KSL. Then, for all $\mathcal{M}, t$:*

$$\mathcal{M}, t \models \psi\ U_{\leq m}\ \phi \Leftrightarrow \mathcal{M}, t \models \bigvee_{i=1}^{m} f_m(\psi, \phi)$$

and

$$\mathcal{M}, t \models \psi\ U_{< m}\ \phi \Leftrightarrow \mathcal{M}, t \models \bigvee_{i=1}^{m-1} f_m(\psi, \phi)$$

where

$$f_m(\psi, \phi) = \begin{cases} \psi \wedge X\ \phi & \text{if } m = 1 \\ \psi \wedge X\ (f_{m-1}(\psi, \phi)) & \text{if } m > 1 \end{cases}$$

The previous result allows us to simplify the model checking algorithm, which thus deals only with the next temporal operator (see Section 4.3).

Example 4.2 [Synchronization] Using the phase-coherence parameter r, it is possible to measure the collective behavior of a system of coupled oscillators. In particular Kuramoto [11] showed that if K is greater than a certain threshold K_c and the oscillators have the same frequency, after a certain amount of time they become perfectly synchronized, i.e. $r = 1$. The property that given a system of N oscillator timed automata with same frequencies, "it becomes perfectly synchronized within 10s", can be specified as:

$$\phi_{psynch} = \Diamond_{\leq 10s}\ r = 1$$

If the oscillators have slightly different frequencies and K is greater than a certain threshold K_c, the phase-coherence parameter r becomes approximately 1. Chosen an ϵ and given a system of N oscillator timed automata with slightly different frequencies, we can specify the property that "within 10s, it becomes synchronized with an approximation of ϵ and, after that, it remains synchronized for at least 5s":

$$\phi^{\epsilon}_{psynch} = \Diamond_{\leq 10s} \Box_{\leq 5s}\, r > 1 - \epsilon$$

Example 4.3 [Locked and drifted oscillators] After a system of oscillators starts to synchronize, the population splits into a partially synchronized state consisting of two groups of oscillators: a synchronized group, called *locked* - that behaves with a frequency locked to the mean of the frequencies distribution - and a desynchronized group, called *drifted*, whose natural frequencies are too extreme to be entrained. In a set of locked oscillators no changes happen to the relative remaining times in two subsequent simulation steps. The property that, given a system of N oscillator timed automata, and given a set $F = \{i \mid 1 \leq i \leq N\}$ of indexes "the oscillators in F become eventually locked within 10s", can be expressed as:

$$\phi^{F}_{locked} = \Diamond_{\leq 10s}\, D^{(1)}.X \bigwedge_{i,j \in F} (d_i - d_j) - (d_i^{(1)} - d_j^{(1)}) = 0$$

4.3 Model checking algorithm

Algorithm 1 shows a high level description of the recursive model checking function `mcKSL`. It takes a KSL formula ϕ - where all the occurrences of the bounded until operator have been translated as shown in Proposition 4.1 - a tuple $\mathcal{M}$, a discretized time (initially 0), and a dignostic trace (initially empty). The function evaluates the formula ϕ over $\mathcal{M}$ and gives a counterexample (a diagnostic trace) whenever the formula is false. The complexity of `mcKSL` is linear with respect to the length of the formula, $O(|\phi|)$. The *addVariables* function allows to store the current state variables $d_i\ :\ \ i = 1, \cdots, N$ in $\mathcal{M}$, when a freeze connective is met during the parsing. For each X connective in the formula, a *nextStep* function is called in order to calculate the evolution of the system using the discrete time Kuramoto model described in Equation (3). The functions *notDiag* and *mergeAnd* simply manage the returning of a correct counterexample, given the results of the recursive calls.

5 Reasoning on synchronization

In this section we briefly report an example of analysis performed by a prototype implementation of our model checker. A demo of the simulator is available at [5]. We generated a system of 40 oscillators by choosing randomly the initial phase synchronizations in the range $(0, 2\pi)$ and frequencies using a Cauchy random variable in the range $(0, 1)$. Figure 3 shows the distribution of frequencies and of phases of each oscillator. We set the coupling strength $k = 1.0$ and we check the ϕ_{locked^F} with $F = \{1, \cdots, 40\}$ and ϕ_{psynch} with $\epsilon = 0.3$. Figure 4 shows the result of the model checking after 10 seconds.

```
function MCKSL($\phi, \mathcal{M}, t$, diagTrace)
                              ▷ Returns (true, $\oslash$) or (false, a diagnostic trace)
    select case
        case $\phi = T$
            return (True, $\oslash$ )
        end case
        case $isAtomic(\phi)$
            return $checkAtomicProp(\phi, \mathcal{M}, t$, diagTrace)
        end case
        case $\phi = \neg\phi_1$
            return $notDiag$(mcKSL($\phi_1, \mathcal{M}, t$, diagTrace), $\phi_1, \mathcal{M}, t$, diagTrace)
        end case
        case $\phi = \phi_1 \wedge \phi_2$
            return $mergeAnd($
                        mcKSL($\phi_1, \mathcal{M}, t$, diagTrace),
                        mcKSL($\phi_2, \mathcal{M}, t$, diagTrace))
        end case
        case $\phi = X\ \phi_1$
            diagTrace $\leftarrow$ $append$(diagTrace, $\phi, \mathcal{M}, t$)
            $\mathcal{M} \leftarrow nextStep(\mathcal{M})$
            return mcKSL ($\phi_1, \mathcal{M}, t+1$, diagTrace)
        end case
        case $\phi = D^{(h)}.\ \phi_1$
            diagTrace $\leftarrow$ $append$(diagTrace, $\phi, \mathcal{M}, t$)
            $\mathcal{M} \leftarrow addVariables(\mathcal{M}, h)$
            return mcKSL ($\phi_1, \mathcal{M}, t$, diagTrace)
        end case
    end select
end function
```

Algorithm 1. mcKSL Algorithm

6 Conclusion and future work

In this paper we have defined a subclass of timed automata, oscillator timed automata, suitable to model biological oscillators. We have specified an interaction semantics, parametric w.r.t. a model of synchronization, and we have instantiated it to the Kuramoto model. The main advantage of the semantic definition is that it does not require changing the structure of the automata modeling the standalone oscillators. We have also introduced a logic, Kuramoto Synchronization Logic (KSL), in order to specify and detect collective synchronization properties in a population of oscillators timed automata. A model checking algorithm has been presented and an example of analysis has been showed.

The main future directions of our work are: 1) extend our logic with *control* operators that allow us to infer how to influence a set of oscillators by adding artificial oscillators, which we can control, in order to satisfy a desired property. 2)

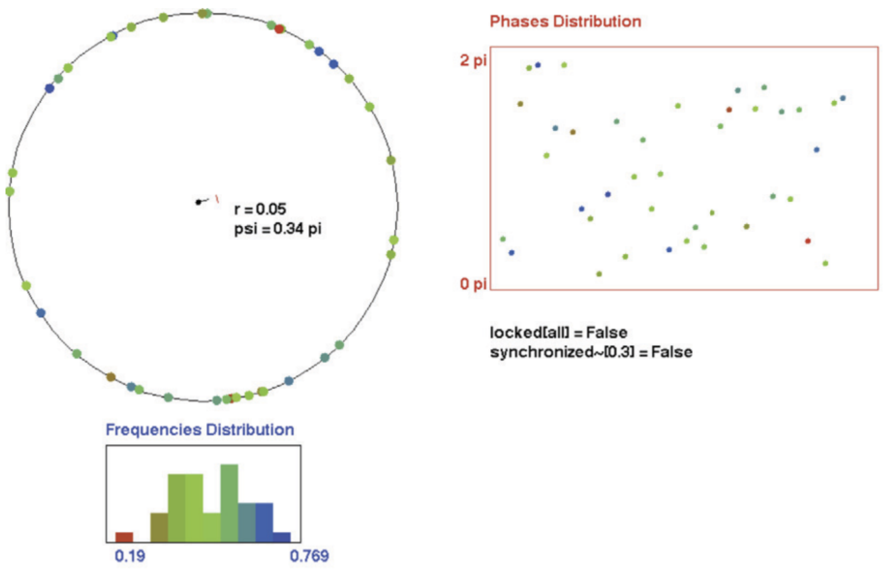

Fig. 3. Initial conditions of the analysis

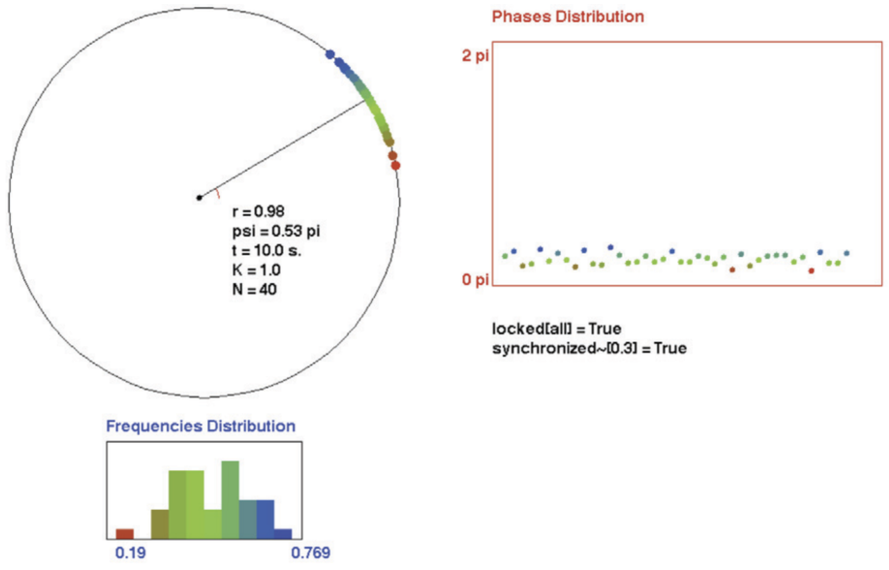

Fig. 4. Verification of properties after 10 seconds

Use hybrid automata, instead of timed automata, to model oscillators and to define the interaction semantics. 3) Fully implement a model checker and simulator for KSL. A prototype is available at [5]. 4) A long term objective: apply the model checking of the extended logic, together with detection techniques already partially available, to control fibrillation in cardiac cells networks (for more details of this line of research see [4]).

Acknowledgement

Research partially supported by the Italian FIRB-MIUR LITBIO: *Laboratory for Interdisciplinary Technologies in Bioinformatics.*

References

[1] Acebron, J., L. Bonilla, C. Pérez Vicente, F. Ritort and R. Spigler, *The Kuramoto model: A simple paradigm for synchronization phenomena*, Rev Mod Phys **77** (2005), pp. 137–185.

[2] Alur, R. and D. L. Dill, *A theory of timed automata*, Theoretical Computer Science **126** (1994), pp. 183–235.

[3] Alur, R. and T. A. Henzinger, *A really temporal logic*, Journal of the ACM **41** (2004), pp. 181–203.

[4] Bartocci, E., F. Corradini, R. Grosu, E. Merelli, S. Smolka and O. Riganelli, "StonyCam: a Formal Framework for Modeling, Analyzing and Regulating Cardiac Myocytes," Lecture Notes in Computer Science, Springer-Verlag, 2008 pp. 493–502.

[5] Bartocci, E., F. Corradini, E. Merelli and L. Tesei, *Prototype of the KSL model checker and simulator.* URL http://cosy.cs.unicam.it/kuramoto/

[6] Buck, J., *Synchronous rhythmic flashing of fireflies II*, The Quarterly Review of Biology **63** (1988), pp. 265–289.

[7] Butcher, J. C., "Numerical methods for ordinary differential equations," John Wiley and Sons, 2003.

[8] Dye, J., *Ionic and synaptic mechanisms underlying a brainstem oscillator: An in vitro study of the pacemaker nucleus of apteronotus*, Journal of Comparative Physiology **168** (1991), pp. 521—-532.

[9] Henzinger, T. A., X. Nicollin, J. Sifakis and S. Yovine, *Symbolic model checking for real-time systems*, Information and Computation **111** (1994), pp. 193–244.

[10] Kitano, H., "Foundations of Systems Biology," MIT Press, 2002.

[11] Kuramoto, Y., *Phase dynamics of weakly unstable periodic structures: Condensed matter and statistical physics*, Progress of theoretical physics **71** (1984), pp. 1182–1196.

[12] Kuramoto, Y., *Collective synchronization of pulse-coupled oscillators and excitable units*, Physica D **50** (1991), pp. 15–30.

[13] Kuramoto, Y., "Chemical Oscillations, Waves, and Turbulence," Springer-Verlag, 2003.

[14] Mirollo, R. E. and S. H. Strogatz, *Synchronization of pulse-coupled biological oscillators*, SIAM Journal of Applied Mathematics **50** (1990), pp. 1645–1662.

[15] Peskin, C. S., "Mathematical Aspects of Heart Physiology," Courant Institute of Mathematical Sciences, New York University, New York, 1975.

[16] Strogatz, S. H., "SYNC, the emerging science of spontaneous order," Hyperion, 2004.

[17] Wang, R., L. Chen and K. Aihara, *Synchronizing a multicellular system by external input: an artificial control strategy*, Bioinformatics **22** (2006), pp. 1775–1781.

[18] Winfree, A. T., *Biological rhythms and the behavior of populations of coupled oscillators*, Journal of Theoretical Biology **16** (1967), pp. 15 – 42.