Detecting Synchronisation of Biological Oscillators by Model Checking

Ezio Bartocci, Flavio Corradini, Emanuela Merelli, Luca Tesei*

School of Sciences and Technology, University of Camerino, Via Madonna delle Carceri 9, 62032 Camerino (MC), Italy

Abstract

We define a subclass of timed automata, called oscillator timed automata, suitable to model biological oscillators. Coupled biological oscillators may synchronise, as emerging behaviour, after a period of time in which they interact through physical or chemical means. We introduce a parametric semantics for their interaction that is general enough to capture the behaviour of different types of oscillators. We instantiate it both to the Kuramoto model, a model of synchronisation based on smooth interaction, and to the Peskin model of pacemaker cells in the heart, a model of synchronisation based on pulse interaction. We also introduce a logic, Biological Oscillators Synchronisation Logic (BOSL), that is able to describe collective synchronisation properties of a population of coupled oscillators. A model checking algorithm is proposed for the defined logic and it is implemented in a model checker. The model checker can be used to detect synchronisation properties of a given population of oscillators. This tool might be the basic step towards the generation of suitable techniques to control and regulate the behaviour of coupled oscillators in order to ensure the reachability of synchronisation.

Keywords: Computational systems biology, Timed automata, Biological oscillators, Model checking, Simulation, Spontaneous synchronisation, Kuramoto model, Peskin model, Pacemaker cells

*Corresponding author

Email addresses: ezio.bartocci@unicam.it (Ezio Bartocci), flavio.corradini@unicam.it (Flavio Corradini), emanuela.merelli@unicam.it (Emanuela Merelli), luca.tesei@unicam.it (Luca Tesei)

1. Introduction

Spontaneous synchronisation happens frequently in nature: pacemaker cells firing, electrons flowing, fireflies flashing, crickets chirping, planets orbiting, neurons firing, menstrual periods synchronising, etc. Every night, along the tidal rivers of Malaysia, thousands of fireflies congregate in the mangroves and flash in unison, without any leader or cue from the environment [1]. In the solar system, gravitational synchrony can eject huge boulders out of the asteroids belt and towards the Earth; the cataclysmic impact of one such meteor is thought to have killed the dinosaurs. Even our bodies are symphonies of rhythm kept alive by the relentless, coordinated firing of thousands of pacemaker cells in our heart [2] or of billions of neurons in our nervous system [3]. Just to mention some of these surprising phenomena analysed by Strogatz in his exciting book [4].

All these phenomena have in common that at their base there are autonomous entities that exhibits a cyclic behaviour: oscillators. Groups of fireflies, planets, or pacemaker cells are collections of oscillators - entities that cycle automatically, that repeat themselves over and over again at more or less regular time intervals. Two or more oscillators are said to be coupled if some physical or chemical process allows them to influence one another. Interactions can be divided in two main types: *smooth-coupled* oscillators interact continuously, while *pulse-coupled* oscillators interact only when an individual firing is observed. Nature uses every available channel to make the oscillators interact: fireflies communicate with light; planets tug on one another with gravity; heart cells pass electrical currents back and forth. The result of these interactions is often synchrony.

These phenomena has been intensively studied by biologists, physicists, mathematicians, astronomers, engineers, sociologists, but beyond the availability of some mathematical models, many questions remain unanswered, as: how exactly do coupled oscillators synchronise themselves, and under what conditions? When is synchronisation impossible and when is it inevitable? What other modes of organisation are to be expected when synchronisation breaks down? Understanding a synchronised collective behaviour is essential in Systems Biology especially for developing methods to control the dynamics of systems and methods to design and modify systems for desired properties [5].

The aim of this work is to provide a general formal framework in which the behaviour of coupled oscillators can be modeled at different levels of

abstraction. Based on this framework, we propose a logic and the relative model checker that allows to prove synchronisation properties, with or without a mathematical model of the biological system. If mathematical results on the occurrence of synchronisation exist, as in Kuramoto [6] or in Peskin [2] models, the model checker will allow to identify the necessary time to reach the synchronisation state. Otherwise, if there are no mathematical results, the model checker allows to analyse the reachability, in a given population of oscillators, of a synchronisation state or of states from which the synchronisation might be achieved relaxing some constraints or giving more information. In the latter case, the model and the model checker can also be used to *validate* hypotheses on parameters derived by observing the real biological system under study.

We believe that this model checking approach may be fundamental in getting insights that are at the basis of understanding the dynamics of coupled oscillators, and we wish to contribute in finding a way for transforming pacemaker cells that are malfunctioning into healthy pacemaker cells, or for controlling cancer nervous cells to turn them into normal nervous cells or to induce the apoptosis in their cell-cycle.

The distributed synchronisation of biological systems is commonly modeled using the theory of coupled oscillators proposed by, among others, Art Winfree [7], Charles S. Peskin [2] and Yoshiki Kuramoto [6]. In this theory, each member of the population is modeled as a phase oscillator running independently at its own frequency. The synchronisation 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 new impulses) or by changing the parameters of individual oscillators. This approach gives rise to an artificial control strategy as it has been proposed by Wang et al. in [8].

The most successful attempt to model distributed synchronisation with smooth interaction has been proposed by Yoshiki Kuramoto. The Kuramoto model, based on Winfree's ideas that mutual synchronisation is a cooperative phenomenon - a temporal analogue of phase transition encountered in statistical physics - is a beautiful and analytically tractable model. A wide description of the Kuramoto model can be found in [9]. On the side of pulse-coupled oscillators, Peskin proposed in [10] the first theory that explains how the different pacemaker cells coordinate their activity so that the whole sinoatrial node fires at the same time.

In this paper we proceed as follows: first we define a subclass of timed

automata, called *oscillator timed automata*, suitable to model oscillators. Biological oscillators can be modeled as very simple automata that exhibit only the oscillation behaviour, but also as detailed automata describing the internal states and events that specify, beyond the oscillation, internal behaviours, possibly connected with the oscillation mechanism. Then, we introduce a non-standard interaction semantics, parametric w.r.t. a model of synchronisation, for describing the parallel composition of a population of oscillator timed automata. To show the generality of the framework we instantiate it to two different running examples of smooth and pulse interactions, i.e. the Kuramoto model and the Peskin model.

Based on such a modelling framework, several analyses could be defined to study different properties of oscillators. In this paper, we provide a logic, called Biological Oscillator Synchronisation Logic (BOSL) by which it is possible to specify and, then, detect synchronisation properties of populations of both smooth and pulse coupled oscillators. We give a model checking algorithm for the logic and we show how to model interesting synchronisation properties as BOSL formulae.

A peculiarity of the logic BOSL is that it is interpreted on states of simulation of the given model. The resulting model checking algorithm can be described as run-time model checking, as it explores the state space simulating the interactions between the oscillator timed automata in a discretised approximated scenario. Thus, we perform a model checking not in the traditional way proving whether a given formula is satisfied on all possible runs of the model, but proving that a particular state - having certain characteristics expressed by a formula - can be reached by simulating successive time steps from an initial state.

We implemented a prototype model checker for BOSL, supporting the Kuramoto model of interaction. As a case study, we use the modelling framework and the logic BOSL for describing pacemaker cells in the heart and for analysing their synchronisation properties.

The paper is organised as follows: Section 2 introduces related works and the two models we use to show our approach: the Kuramoto model and the Peskin model. Section 3 recalls timed automata, defines oscillator timed automata, and specifies the interaction semantics. Section 4 introduces the Biological Oscillator Synchronisation Logic (BOSL) and its model checking algorithm presenting the case study at the end. Section 5 concludes outlining some directions for future work. A preliminary version of this work appeared in [11].

2. Background

In this section we first present the literature, in the field of biological oscillators, that inspired our work. Then we introduce the basic concepts of phase oscillators and their mathematical model. Finally, we describe in detail the two models of synchronisation we use as running examples: the Kuramoto model for smooth-coupled oscillators and the Peskin model for pulse-coupled oscillators.

2.1. Related work

During the last decades, several mathematical models have been proposed to study the spontaneous synchronisation phenomena in a population of biological coupled oscillators [4]. These models have been inspired by real biological systems, ranging from the mutual synchronisation 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 synchronisation could be performed both through smooth interactions and through episodic impulses.

For the first case, in which the interactions between oscillators are smooth, a first approach was proposed by Winfree [7] that introduced a model of nearly identical, weakly coupled limit-cycle oscillators. Using numerical simulation, he discovered that in this loose-coupling hypothesis the system behaves incoherently, with each oscillator running at its natural frequency. He also found that, as the coupling is increased, the unsynchronised incoherence continues until a certain threshold, when a group of oscillators jump suddenly into synchrony.

Starting from Winfree's results and assumptions, Kuramoto began to work with collective synchronisation phenomena and he proposed a refined model [6, 12] providing some analytical tools in order to render the problem more tractable and the synchronisation measurable.

The second 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 [2] who proposed a model of the mutual synchronisation of sinoatrial 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 [13] demonstrated that the conjecture holds

for all N. Peskin also conjectured that synchronisation would occur even if the oscillators were not quite identical, but that problem still remains open.

Our model is a candidate to treat both smooth and pulse oscillators. In the following sections we introduce phase oscillators and we briefly report the results of the mentioned research in order to introduce the concepts who inspired us to construct the framework and the results on which we relied on to define the logic for synchronisation detection.

2.2. Phase oscillators

Generally speaking, we aim to describe 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 visualised 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 behaviour of the others. This can be viewed as a process of collective synchronisation that ends up, under certain conditions, in a total synchronous behaviour. In the metaphor of the points moving on the circle, when the system becomes synchronised the points move around in sync, meaning that the phase differences remain constant. Under certain circumstances these differences are also null.

In case of smooth coupled oscillators the adapting process is continuous, i.e. the speed $\dot{\theta}_i$ and the acceleration $\ddot{\theta}_i$ are continuously updated depending on the state of the other oscillators. In case of pulse coupled oscillators there is a sudden discrete change in the state of an oscillator when it perceives another oscillator emitting a particular signal, for instance firefly flashing or pacemaker cell firing.

2.3. Kuramoto model

The Kuramoto model of synchronisation [14] describes the evolution of a population of N smooth coupled phase oscillators. By Kuramoto, the evolution of the interacting i-th oscillator is given by the following equation:

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

where K is a parameter of the system representing the coupling strength, which depends on the type of interaction. A primary basic condition to

the possibility of synchronisation 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 of the distribution and ω_0 is the median.

In his analysis [14], Kuramoto provided a measure of synchronisation 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 oscillators and it is a convenient measure of the extent of synchronisation in the limit $N \to \infty$ and $t \to \infty$. If all oscillators are in sync, then $r = 1$ when all the frequencies ω_i 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 desynchronised if the value of the coupling strength K is below a critical threshold K_c. Above this value, the population starts splitting into a partially synchronised state consisting of two groups of oscillators: a synchronised group that contributes to the order parameter r, and a desynchronised group whose natural frequencies lie in the tails of the distribution $g(\omega)$ and are too extreme to be entrained. The higher is the value of K the more are the oscillators recruited into the synchronised group, with r growing accordingly.

A special case is when all oscillators have the same natural frequency. In this situation the interactions will take the population to a steady state in which there are no more interactions. In the majority of cases the oscillators are all perfectly synchronised. In other cases, for instance if the difference of

phases are π - yielding a null interaction, they are *locked*, i.e. they move at the same speed but there are differences of phase that remain constant over time.

2.4. Peskin model

The heart beat originates in the sinoatrial node, a region of cells which have the capability of depolarising spontaneously towards a threshold, firing, and then recovering. Peskin in [10] proposed the first theory that explained how the different cells coordinate their activity so that the whole sinoatrial node fires at the same frequency and (except for conduction delays) in phase. His main hypothesis was that cells behave as a population of weakly pulse-coupled oscillators, in which synchrony emerges as a consequence of the interaction and in which the overall frequency is a property of the population of cells, rather than any single cell. He modeled the pacemaker cells as a fully-connected network of N identical (same frequency) pulsed-coupled oscillators, each characterised by a voltage-like state variable x_i, subject to the following dynamics:

$$\dot{x}_i = S_0 - \gamma x_i, \quad 0 \leq x_i \leq 1, i = 1, \ldots, N \tag{3}$$

where S_0 and γ, $S_0 > |\gamma|$, $\gamma \neq 0$, are the intrinsic properties of the oscillator. When $x_i = 1$, the i-th oscillator "fires" and x_i jumps back to zero. The oscillators are assumed to interact by a simple form of pulse coupling: when a given oscillator fires, it pulls all the other oscillators up by an amount ε, or pulls them up to firing, whichever is less. That is:

$$x_i(t) = 1 \Rightarrow x_j(t^+) = min(1, x_j(t) + \varepsilon) \quad \forall j \neq i \tag{4}$$

Peskin in [10] conjectured that cells would all end up firing in unison, no matter how they started. He gave a proof for $N = 2$ oscillators; it was later demonstrated by Mirollo and Strogatz in [15] that the conjecture holds for all N. Peskin also conjectured that synchronisation would occur even if the oscillators were not quite identical, but that problem remains open. Peskin's model has been used as a caricature of coupled neurons [16, 17, 18] by including synaptic delays, refractory periods, inhibition, and local coupling.

3. Automata model

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

3.1. Timed automata

Timed automata [19] are an established formalism for modelling 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 of 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 $\mathbb{R}^{\geq 0}$. Clocks measure time as it elapses. All clocks of a given system advance at the same rate: when increasing, they can be viewed as functions of 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 non-negative real 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 last action/event that reset the clock. 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 $x \in \mathcal{X}\backslash\gamma$.

The timed behaviour 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 $(Q, \Sigma, \mathcal{E}, q_0, \mathcal{X}, Inv)$, 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*

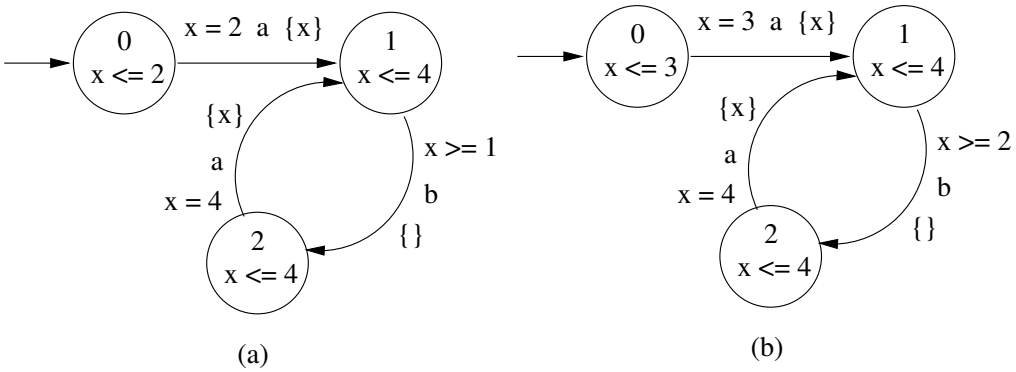

Figure 1: Two similar oscillator timed automata on distinguished action a with the same period 4 but different initial delays and different behaviours on action b.

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, a variant of the original model introduced in [20], that are the most common in the modelling and verification tools. They incorporate a notion of urgency, 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 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 0 is $x <= 2$. There is an edge from state 0 to state 1 with clock constraint $x = 2$, label a and reset set $\{x\}$.

The semantics of a timed automaton $T = (Q, \Sigma, \mathcal{E}, q_0, \mathcal{X}, Inv)$, is a labelled 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 lets 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 describes 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 target 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 behaviour of the automaton in Figure 1(a) is $r_{ex} = (0, [x = 0]) \xrightarrow{2} (0, [x = 2]) \xrightarrow{a} (1, [x = 0]) \xrightarrow{1.2} (1, [x = 1.2]) \xrightarrow{b} (2, [x = 1.2]) \dots$

Let $T = (Q, \Sigma, \mathcal{E}, q_0, \mathcal{X}, Inv)$ be a timed automaton and let r be an *infinite* derivation of $S(T)$, $r = s_0 \xrightarrow{l_0} s_1 \xrightarrow{l_1} \cdots$ 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[1]:

$$t_{-1} = 0$$
$$t_{i+1} = t_i + \begin{cases} 0 & \text{if } l_i \in \Sigma \\ l_i & \text{otherwise} \end{cases}$$

  We say that r is *divergent* if for every $M \in \mathbb{R}^{\geq 0}$ there exists $i \in \mathbb{N}$ such that $t_i > M$

- The *label sequence* of r is the sequence of the transitions occurred during r, including the elapsed times from the beginning of the derivation, i.e., from the initial state: $(l_0, t_0)(l_1, t_1) \cdots$

- 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 $t_{-1} = 0, t_0 = 2, t_1 = 2, t_2 = 3.2, t_3 = 3.2, \dots$ The label sequence is $(2, 2)(a, 2)(1.2, 3.2)(b, 3.2) \cdots$ The action sequence is $(a, 2)(b, 3.2) \cdots$ The b-sequence is $(b, 3.2) \cdots$

[1]Index 0 is associated to the first time t_0 of the first move of the transition system. So the initial time 0 is indexed by -1.

3.2. Oscillator timed automata

We want to identify a subclass of timed automata that are suitable to represent phase oscillators. In the following we discuss the main issues, both technical and conceptual, that we considered in order to characterise oscillator timed automata and then we formalise them in Definition 3.2.

First, it is more convenient, in a timed setting, to represent the parameters of the oscillators with time values instead of angular values such as angular speed or initial phase. Thus, we represent the frequency ω by a *period* of oscillation p and the initial phase θ_0 ($0 \leq \theta_0 < 2\pi$) by an *initial delay* ϑ_0 ($0 \leq \vartheta_0 < p$). They are simply related as follows: $\omega = \frac{2\pi}{p}$ and $\theta_0 = \frac{2\pi}{p}\vartheta_0$.

From the point of view of the modelling formalism, we decided to use the full power of timed automata to allow the specification of biological oscillators at different levels of abstraction. This means that, on one hand, an oscillator can be described with a very simple automaton that represents the mere oscillation (with the frequency and the initial phase) without other details. On the other hand, using locations, actions and non-determinism of timed automata one can specify, using the available biological information, a more complex system that exhibits an oscillating behaviour among other "internal" behaviours and states. An example of this approach is given in Section 4.5 where we describe pacemaker cells with information about the various phases they go through in their cycle. Having a detailed model of an oscillator allows to perform analyses and verifications depending not only on the "external" observation of the oscillation, but also on the "internal" aspects of the oscillator.

The degree of freedom in modelling, though, must be balanced with a strict condition on what really defines an oscillator, i.e. the fact that a cyclic behaviour repeats regularly over time. To fix this, we should identify an observable event, in the run of a timed automaton, that represents this regular cycle. The natural choice is a certain *distinguished* action that must repeat regularly on every trace of the automaton, no matter which locations are traversed and which other actions are performed. This imposes a certain degree of determinism on the automaton. Clock constraints with equalities in timed automata allow us to be "punctual", i.e. to identify precise points in time in which an event have to occur. However, as we discuss below, this punctuality could be relaxed in order to get a more flexible framework.

Another aspect to be considered is the initial condition, i.e. the initial phase of the oscillator. This parameter is sometimes crucial for the possibility of the oscillator to synchronise when it is inserted in a population of coupled

oscillators[2]. Thus, the initial phase must be specified properly and precisely if we want to identify a correct initial state for the automaton. Also in this case we impose determinism and punctuality, requiring the automaton to fire its first distinguished action precisely at the given initial delay. This has the advantage to simplify the definition of the initial state of interaction, as we discuss in Section 3.3.2.

Definition 3.2. *A timed automaton T is called* oscillator timed automaton *on a* distinguished action $a \in \Sigma$ *with* period $p \in \mathbb{R}^{>0}$ *and* initial delay ϑ, $0 \leq \vartheta < p$, *if and only if the two following conditions hold:*

1. *for each infinite divergent derivation r of $S(T)$ the a-sequence of r is an infinite sequence of the form $(a, \vartheta)(a, 1 \cdot p + \vartheta)(a, 2 \cdot p + \vartheta)(a, 3 \cdot p + \vartheta) \cdots$*
2. *every finite derivation of $S(T)$ is a prefix of an infinite divergent derivation r of $S(T)$*

This definition identifies as oscillator timed automata those that start the oscillation performing their distinguished action a for the first time exactly at their given initial delay ϑ and then regularly repeat the distinguished action every period p from the first action on (condition (1)). Note that the distinguished action a is required not to occur between any two occurrences separated by the period p, otherwise the a-sequence would contain that occurrence of a, which would occur before the next period and would falsify the condition.

Condition (2) imposes that oscillator timed automata have no dying paths, i.e. paths ending in a state where time can not proceed. This is equivalent to consider only the divergent behaviours of the automata, i.e. we neglect, as usually done in the context of timed automata, those infinite derivations (called Zeno derivations) in the transition system $S(T)$ of an automaton T in which time converges due to a choice of a convergent succession of delays δ in $\mathbb{R}^{\geq 0}$. This condition is needed for technical reasons that will become more clear in Section 3.3.1. Informally, we want to have the possibility to make an oscillator timed automaton proceed at fixed small time steps. For doing this we must be guaranteed that every piece of derivation we make

[2]For instance, if two smooth-coupled oscillators interacting with the Kuramoto model have initial phases that differ of π their interaction will always be null, thus they will not synchronise.

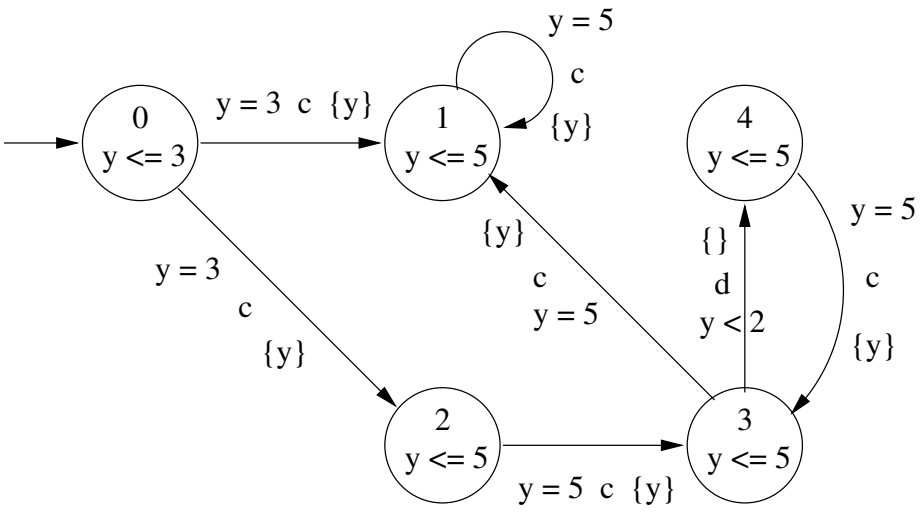

Figure 2: A more complex oscillator timed automaton.

takes us to a state in which the derivation can continue, no matter how the non-determinism was resolved.

Figure 1(a) shows an oscillator timed automaton on the distinguished action a with period 4 and initial delay 2, while the automaton in Figure 1(b) has the same distinguished action and the same period, but initial delay 3. Note that the two automata can represent two oscillators with the same frequency and different initial phase. Note also that the non-oscillating behaviours of the two automata are different in the sense that the non-distinguished action b can occur, between any two occurrences of a, with different time constraints in the two automata.

In Figure 2 we show a slightly more complex automaton which is an oscillator timed automaton on the distinguished action c with period 5 and initial delay 3. Note that there is non-determinism on the choice of the initial c-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. As we mentioned above, the choice of the distinguished action identifies the particular observation with which an external observer recognises the oscillating behaviour.

We want to remark that our definition of oscillator timed automata could be too strong in situations in which a high punctuality of time constraints is not required. In these cases, weaker definitions - for instance using less strict acceptance conditions for the time trajectories of the automata as in [21] - can be used without changing significantly the interaction semantics and the model checking approach presented in Section 4.

Finally, an important issue about Definition 3.2 is the possibility to automatically check if a given timed automaton is an oscillator timed automaton on a certain distinguished action a with a certain period p and initial delay ϑ. Concerning condition (1), one can use classical model checking on timed automata to assure that the initial condition holds and that the oscillation occurs forever at precise times. For the former it is sufficient to express a reachability requirement on all paths, while the latter can be expressed as a classical invariance property requiring that whenever the distinguished action a occurs then, on all possible paths, after exactly p time units another a occurs, and in the meantime it does not occur. As the properties we have to check depend on the actions labelling the edges of the automaton, in particular the distinguished action a, a suitable temporal logic to express these requirements is ATCTL [22]. An ATCTL formula can be mapped to a TCTL formula [20], which can be verified with the model checking tool KRONOS [23].

For what concerns condition (2), in [20] it is given a fix-point algorithm to compute if a given timed automaton does not contain Zeno states, i.e. states from which time can not advance towards divergence. This algorithm is implemented within the KRONOS tool and, thus, also this condition can be checked.

3.3. Interacting oscillator timed automata

Standard synchronisation between timed automata, either classical collective action-based synchronisation [19, 23] or point-to-point channel communication in a network of timed automata [24, 25], is not suitable for representing the synchronisation that takes place among biological oscillators. The kind of synchronisation we deal with is an *emerging behaviour* and the means of communication can not be easily represented by channels or handshakes because they are actually perturbations of the normal behaviour of the single oscillator due to physical or chemical interactions with a dynamic environment.

In this section we propose a way to describe the interaction among several oscillator timed automata in order to possibly obtain synchronisation as emerging behaviour. The semantics of interaction we give here is non-standard and is parametric with respect to the model of synchronisation. After the parametric definition we introduce two particular instances: one based on smooth interaction using the Kuramoto model and another based on pulse interaction using the Peskin model.

Suppose we are given N oscillator timed automata $T_1, \ldots, T_N$ on distinguished actions $a_1, a_2, \ldots, a_N$ with periods $p_1, \ldots, p_N$ and 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 $(q, \psi, \gamma \cup \{\overline{x}_i\}, a_i, q')$. Note that $\overline{x}_i$ needs to be added because a clock functioning in this way may not exist in T_i.

The previous transformation ensures that, after a proper initialisation, $\overline{x}_i$ 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 initialisation process is described in Section 3.3.2.

3.3.1. Steps of activity

In order to describe the interaction semantics we need to define the *step of activity* of a single automaton. This step is intended as a certain small time interval of simulation of an oscillator dynamics. The length of the step will be determined by the discretisation of the model, as we describe later, and by the interaction with other oscillators in the population under analysis, following the interaction semantics assigned for the given type of oscillators.

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 $(q, \nu) = (q^0, \nu^0) \xrightarrow{l_0} (q^1, \nu^1) \xrightarrow{l_1} \cdots \xrightarrow{l_{n-1}} (q^n, \nu^n) = (q', \nu')$ 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'). However, here the time count starts at zero from the beginning of the considered derivation, thus the times t_j in the time sequence are all in the interval $[0, \Delta]$. 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 behaviour. 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 (possibly empty) 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. The thesis follows directly from Definition 3.2. □

Consider the automaton shown in Figure 1(a). A possible step of activity of length 3 from the initial state is $(0, [x = 0]) \xrightarrow[3]{2\ a\ 1} (1, [x = 1])$ corresponding to the derivation $(0, [x = 0]) \xrightarrow{2} (0, [x = 2]) \xrightarrow{a} (1, [x = 0]) \xrightarrow{1} (1, [x = 1])$ whose action sequence is $(a, 2)$.

Consider the automaton shown in Figure 1(b). A possible step of activity of length 3 from the initial state is $(0, [x = 0]) \xrightarrow[3]{3\ a} (1, [x = 0])$ corresponding to the derivation $(0, [x = 0]) \xrightarrow{3} (0, [x = 3]) \xrightarrow{a} (1, [x = 0])$ whose action sequence is $(a, 3)$.

3.3.2. Initialisation

The initialisation 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 $S(T_i)$.

Proposition 3.4. *Given N oscillator timed automata T_i, $i = 1, \ldots, N$, on distinguished actions a_i, with periods p_i and initial delays ϑ_i, we can effectively determine, starting from the initial states of every $S(T_i)$, a tuple $\langle \overline{s}_0^1, \overline{s}_0^2, \ldots, \overline{s}_0^N \rangle$ such that for all i:*

- $\overline{s}_0^i \in S(T_i)$
- *T_i fired at least once the distinguished action a_i exactly at the assigned initial delay ϑ_i*
- *the time elapsed during the derivation from the initial states of $S(T_i)$ to $\overline{s}_0^i$ is exactly $max(\{\vartheta_1, \ldots, \vartheta_N\})$*

Proof. Let $\vartheta_{\max} = \max(\{\vartheta_1, \ldots, \vartheta_N\})$. By Proposition 3.3 every automaton T_i such that $\vartheta_i < \vartheta_{\max}$ can perform, from its initial state, a step of activity of length $\vartheta_{\max}$: $s_0^i \xrightarrow[\vartheta_{\max}]{\ell} \overline{s}_0^i$. By the properties stated in Definition 3.2, state $\overline{s}_0^i$ satisfies all the conditions of the thesis.

For automata T_j whose initial delays are equal to $\vartheta_{\max}$ we can perform the same step as above and possibly have to add a final transition $\xrightarrow{a_j} \overline{s}_0^j$ if the reached state did not result from the firing of the distinguished action. Again, the possibility to add this transition is guaranteed by the conditions in Definition 3.2. □

Note that, during the procedure described in the proof, the automata whose initial delays were less than the maximum initial delay 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 synchronisation 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.

Note also that the tuple $\langle \overline{s}_0^1, \overline{s}_0^2, \ldots, \overline{s}_0^N \rangle$ is not uniquely defined due to possible non-determinism in the performed steps of activity.

Consider the two automata in Figure 1 numbered 1 (a) and 2 (b). Using the procedure suggested in the proof, a candidate tuple for initiating their interaction is $\langle(1, [x = 1, \overline{x}_1 = 1]), (1, [x = 0, \overline{x}_2 = 0])\rangle$.

3.3.3. Interaction semantics

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

Our objective is to approximate the continuous dynamics of the *interacting* automata $T_1, T_2, \ldots, T_N$ without changing the structure of the standalone automata T_i's. Other approaches [26, 27, 28] exist to approximate dynamic behaviours typical of hybrid automata [29], or in general of hybrid systems, by using timed automata. However, they focus on developing techniques for performing classical (approximated) verifications on such systems, for which even the simplest verification problems are often undecidable. In this paper, instead, we define a non-standard semantics for the parallel composition of timed automata in order to describe a collective process of "emerging synchronisation" of them.

First, to approximate the continuous dynamics of the interaction between the oscillators - for instance the Kuramoto interaction described by Equation (1) or the Peskin interaction described by Equation (4) - we introduce in our model a concept of discretisation. In more detail, we consider a small *fixed* time interval δt as the pace at which all the population of oscillator timed automata evolves over time. The value of δt has to be considered a global parameter depending on the precision required by the application. Interactions take place at every interval of length δt. Note that this does not mean that we use a discrete time domain: the underlying time domain is still $\mathbb{R}^{\geq 0}$, but the interactions are not continuous, occurring every δt time units.

Then, at each step, we use steps of activities defined in Section 3.3.1 to make each standalone automaton T_i advance of an amount of time Δ_i which is different, in general, from the global δt pace. More precisely, it is the model of synchronisation, by means of the interaction function $\mathcal{I}$, that decides the length of Δ_i. In this way we can obtain a perturbed behaviour of the global system, $T_1, T_2, \ldots, T_N$ in parallel, due to the interaction. This is achieved by considering the time of the external observer (δt) as the *global* time, while the N different Δ_i's are considered as *relative* times, i.e. valid only internally for each oscillator timed automaton. If, according to the interaction, T_i has to decelerate and T_j has to accelerate during the current slice of global time δt,

then T_i performs a step of activity whose duration Δ_i is shorter than δt, while T_j performs a longer Δ_j. The global observer detects perturbed behaviours because the timestamps of the actions performed by each interacting T_i are rescaled to the magnitude of the global time slice δt and merged together to form a unique timed trace.

Let us formalise this non-standard semantics of parallel composition. First, we 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:

$$\frac{\forall i = 1, \ldots, N \; \Delta_i = \mathcal{I}(i, \delta t, s_1, s_2, \ldots, s_N) \quad s_i \xrightarrow[\Delta_i]{\ell_i} s'_i \quad \lambda^i = \mathcal{SC}(\mathcal{A}(\ell_i), \Delta_i, \delta t)}{\langle s_1, s_2, \ldots, s_N \rangle \xrightarrow[\delta t]{\lambda^1, \lambda^2, \ldots, \lambda^N} \langle s'_1, s'_2, \ldots, s'_N \rangle}$$

where the rescaling function $\mathcal{SC}$ is defined as follows:

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

Note that, in the rule above, the rescaling function is called with the proper current time slice Δ_i of each automaton i. Thus, every action time t_j in the given action sequence - which is in the interval $[0, \Delta_i]$ (see Section 3.3.1) - is rescaled w.r.t. the ratio $\delta t / \Delta_i$, which is the one to be considered for automaton i.

Moreover, note that the resulting state s'_i of each step of activity $s_i \xrightarrow[\Delta_i]{\ell_i} s'_i$ is determined non-deterministically by the choices made in the subderivation represented by ℓ_i. In particular, there may be more than one ending state s'_i if the automaton is non-deterministic. However, Proposition 3.3 assures that from each possible s'_i the computation can continue and diverge.

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

$$\rho = \langle \overline{s}_0^1, \overline{s}_0^2, \ldots, \overline{s}_0^N \rangle \xrightarrow[\delta t]{\lambda_0^1, \lambda_0^2, \ldots, \lambda_0^N} \langle s_1^1, s_1^2, \ldots, s_1^N \rangle \xrightarrow[\delta t]{\lambda_1^1, \lambda_1^2, \ldots, \lambda_1^N} \cdots$$

Then, 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, in order to compute incrementally the timestamps

from the beginning of the trace, 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*

$$\rho = \langle \overline{s}_0^1, \overline{s}_0^2, \ldots, \overline{s}_0^N\rangle \xrightarrow[\delta t]{\lambda_0^1,\lambda_0^2,\ldots,\lambda_0^N} \langle s_1^1, s_1^2, \ldots, s_1^N\rangle \xrightarrow[\delta t]{\lambda_1^1,\lambda_1^2,\ldots,\lambda_1^N} \cdots$$

is the following action sequence:

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

The observable behaviours *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 behaviours. Projecting a trace on distinguished actions only allows to observe the synchronisation process if, eventually, all the distinguished actions occur at the same times.

Consider again the two automata in Figure 1 and the initial tuple calculated in the previous section: $\langle(1,[x=1,\overline{x}_1=1]),(1,[x=0,\overline{x}_2=0])\rangle$.

Suppose that $\delta t = 0.5$, $\mathcal{I}(1,0.5,(1,[x=1,\overline{x}_1=1]),(1,[x=0,\overline{x}_2=0])) = 0.6$ and $\mathcal{I}(2,0.5,(1,[x=1,\overline{x}_1=1]),(1,[x=0,\overline{x}_2=0])) = 0.35$.

Then, according to the interaction function $\mathcal{I}$, T_1 has to accelerate performing a step of activity of length 0.6 instead of the "real" length $\delta t = 0.5$. On the contrary, T_2 has to decelerate performing a step of activity of length 0.35. Applying the rule we have to find the successor states after the steps of activity. Two possibilities are $(1,[x=1,\overline{x}_1=1]) \xrightarrow[0.6]{0.4\ b\ 0.2} (2,[x=1.6,\overline{x}_1=1.6])$ and $(1,[x=0,\overline{x}_2=0]) \xrightarrow[0.35]{0.2\ 0.15} (1,[x=0.35,\overline{x}_2=0.35])$ whose action sequences are $(b,0.4)$ and empty, respectively. The rescaling of $(b,0.4)$ has to be done according to the ratio $\delta t/\Delta_1$, i.e. $0.5/0.6$. Thus, $\lambda^1 = (b, 0.4\cdot 0.5/0.6) = (b,0.33)$, i.e. the external observer see the timestamp 0.33 associated to b, different from the "internal" relative timestamp 0.4 that the standalone T_1 would exhibit.

3.3.4. Kuramoto interaction

In this section we instantiate the interaction semantics given above to the Kuramoto model of synchronisation introduced in Section 2.3.

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 δt. 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 discretisation of the model (see Section 3.3), during the time slice δt the i-th oscillator has to move forward, in the circle representation, of an angle $(\omega_i + \alpha_i)\delta t$, 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 $(\omega_i + \alpha_i)\delta t = \omega_i\Delta_i$. Thus, $\Delta_i = \delta t + \frac{\alpha_i\delta t}{\omega_i}$, which is the value of $\mathcal{I}(i, \delta t, s_1, s_2, \ldots, s_N)$.

3.3.5. Peskin interaction

In this section we show another possible instance of the interaction semantics in which the oscillators are pulse coupled. The model is that of Peskin introduced in Section 2.4.

When considering pulse interactions, if in a time slice δt none of the oscillators performs its distinguished action, which we use to represent the "pulse event" observed by the others, every oscillator goes ahead with its

normal behaviour without perturbations. Thus, we only need to specify the interaction when *at least one* of the oscillator is going to fire in the current δt.

Recall the dynamics of interactions of pacemaker cells introduced in Section 2.4. To frame Equation (4) in our model we scale the $[0, 1]$ interval of the cell voltage to the interval $[0, 2\pi]$. The initial voltage v_i^0 of oscillator i becomes, in our model, its initial phase as $\theta_i^0 = v_i \cdot 2\pi$. The frequency of the oscillator is calculated, using the relation specified in the previous section, from its period p_i, which can be determined using Equation (3) as follows:

$$p_i = \int_0^1 \frac{1}{S_0 - \gamma x_i} dx_i = \frac{1}{\gamma} \ln \left| \frac{S_0}{S_0 - \gamma} \right|$$

with the conditions that $|\gamma| < S_0$ and $\gamma \neq 0$.

The parameter ε of Equation (4), a small pulse of voltage, needs to be transformed into the corresponding small pulse of time ε_t as $\varepsilon_t = p_i \varepsilon$.

Let u_i, for all i, be the time elapsed since the last occurrence of the distinguished action a_i, as described in the previous section. We consider the vector $\mathbf{v}$ with components $v_j = p_j - (u_j + \delta t), j = 1, \ldots N$. If all v_j are positive, then no oscillator is going to pulse in this δt and the value of $\mathcal{I}(i, \delta t, s_1, s_2, \ldots, s_N)$ is exactly δt for all i. Otherwise, let j be the index of the minimum value in vector $\mathbf{v}$[3]. In this case we know that oscillator j is the first that is going to fire in this δt. Then, according to Equation (4) and to the rescaling in the time domain we discussed above, we get:

$$\mathcal{I}(i, \delta t, s_1, s_2, \ldots, s_N) = \begin{cases} \delta t + \varepsilon_t & \text{if} \quad i \neq j \wedge v_i > 0 \\ \delta t + \min(\varepsilon_t, |v_j - v_i|) & \text{if} \quad i \neq j \wedge v_i \leq 0 \\ \delta t & \text{if} \quad i = j \end{cases}$$

The second case corresponds to the situation in which oscillator i is close to fire in this δt but it is preempted by oscillator j. Then we bring oscillator i to perfectly synchronise with oscillator j as required in Equation (4), unless the time pulse ε_t is shorter than the time leap required to perfectly synchronise i and j.

[3]If more than one component has the minimum value, then we take anyone of them.

4. Logic for biological oscillators

In this section, we introduce a logic, Biological Oscillator Synchronisation Logic (BOSL), in order to specify and, thus, detect collective synchronisation properties of a population of coupled oscillators modeled as oscillator timed automata. We provide the syntax, the semantics and a model checking algorithm for BOSL. At the end of the section we present a case study showing how oscillator timed automata and BOSL can be used to analyse a real scenario as the synchronisation of pacemaker cells.

4.1. State model of BOSL

In BOSL the flow of time is a succession of consecutive time intervals of the *fixed* pace δt, as we introduced in Section 3.3. As usual in discretised scenarios, the shorter is δt the more accurate is the synchronisation detection and the more complex is the computation. To simplify the notation, we use natural numbers to denote time steps: $t = 0$ is the initial time and time step $t > 0$ corresponds to the real time $t \cdot \delta t$.

At each time step we consider an underlying population of oscillator timed automata $T_1, \ldots, T_N$ to which we associate a state

$$\mathcal{M} = (\langle s_1, \ldots, s_N \rangle, \mathcal{D}, \mathcal{D}^{(*)}, \mathcal{C}, \mathcal{P})$$

Each s_i is a state of $S(T_i)$ in which the transformation described in Section 3.3 has been applied in order to introduce the special clocks $\overline{x}_i$. $\mathcal{D} = \{d_1, \cdots, d_n\}$ is a vector of state variables whose values are the *remaining times*, for the oscillator timed automata we are considering, to accomplish their distinguished actions. The values of each d_i can be easily inferred from the state $s_i = (q, \nu)$ as $p_i - \nu(\overline{x}_i)$. $\mathcal{D}^{(*)}$ has to be intended as a store. It is a set of vectors of constants whose values are *stored remaining times* used to save the values of the state variables d_i at certain time steps in order to compare them with successive values of the state variables or with other values in the state. These stored values are needed for correctly defining the semantics of a special operator of the logic that *freezes* the values and use them in its subformula. $\mathcal{C}$ is a vector of state constants, usually representing the parameters of the interaction model. $\mathcal{P}$ is a vector of *calculated* state variables whose values depend on the variables in $\mathcal{D}$ according to the particular interaction semantics of the underlying oscillator timed automata. For instance, for the Kuramoto interaction (see Section 3.3.4) $\mathcal{C} = \{K\}$ where $K \in \mathbb{R}$ is the *interaction constant* between the oscillators automata, and $\mathcal{P} = \{r\}$ where r

is the *phase coherence* calculated at any time step using Equation (2). In the case of Peskin interaction $\mathcal{C} = \{\varepsilon_t, \overline{\gamma}, \overline{S_0}\}$ where ε_t is the small time pulse parameter used to accelerate the oscillators when they perceive a pulse event from another oscillator, and $\overline{\gamma}$, $\overline{S_0}$ are vectors containing the intrinsic properties of the N oscillators (see Sections 2.4 and 3.3.5). $\mathcal{P}$ is empty in this case.

An initial state, denoted $\mathcal{M}_0$, is such that the tuple $\langle \overline{s}_0^1, \overline{s}_0^2, \ldots, \overline{s}_0^N \rangle$ calculated as described in Proposition 3.4, the variables in $\mathcal{D}$ are assigned consequently, the values of $\mathcal{P}$ are calculated accordingly to the interaction model and $\mathcal{D}^{(*)}$ is empty.

Let us describe how a new state $\mathcal{M}' = (\langle s'_1, \ldots, s'_N \rangle, \mathcal{D}', \mathcal{D}^{(*)}, \mathcal{C}, \mathcal{P}')$ at time step $t+1$ is obtained from a state $\mathcal{M} = (\langle s_1, \ldots, s_N \rangle, \mathcal{D}, \mathcal{D}^{(*)}, \mathcal{C}, \mathcal{P})$ at time step t.

First, we perform a step of the progress relation described in Section 3.3.3:

$$\langle s_1, s_2, \ldots, s_N \rangle \xrightarrow[\delta t]{\lambda^1, \lambda^2, \ldots, \lambda^N} \langle s'_1, s'_2, \ldots, s'_N \rangle$$

This means that the particular interaction function $\mathcal{I}$ of the underlying oscillators has to be used and one of the possible non-deterministically chosen subderivations are performed internally by the automata.

Then, according to the given semantics, the new values of each state variable d'_i of $\mathcal{D}'$ are

$$d'_i = \begin{cases} d_i - \Delta_i & \text{if} \quad \Delta_i \leq d_i \\ p_i - (\Delta_i - d_i) & \text{if} \quad \Delta_i > d_i \end{cases}$$

where $\Delta_i = \mathcal{I}(i, \delta t, s_1, s_2, \ldots, s_N)$ is the time calculated by the interaction function $\mathcal{I}$ - instantiated to the particular kind of interaction considered (see, for instance, Section 3.3.4 and Section 3.3.5) - for the current step of activity of oscillator i in the underlying population of oscillator timed automata, as described in Section 3.3.

Finally, we have to re-compute the values $\mathcal{P}'$ of calculated state variables, using the new values of $\mathcal{D}'$, according to the equations defining them.

Differently from classical state models that are Kripke structures, the state space of BOSL is a forest of infinite trees whose nodes are pairs $(\mathcal{M}, t)$. The root of each tree is a node $(\mathcal{M}_0, 0)$, where $\mathcal{M}_0$ is one of the possible initial states[4]. The child nodes of a node $(\mathcal{M}, t)$ are all the possible nodes $(\mathcal{M}', t+1)$

[4]The number of trees in the forest is given by the (finite) number of possible initial

that can be obtained by the procedure described above. The branching size is given by the (finite) number of possible different non-deterministic choices that can be made in the step of the progress relation $\xrightarrow[\delta t]{\lambda^1,\lambda^2,\ldots,\lambda^N}$.

However, as we shall see in the following, the infiniteness of the state space is not a problem for the model checking of a given BOSL formula. Informally, since the properties observed by BOSL are independent from how the non-determinism is resolved, then the model checker can chose any tree in the forest and any of the branching when it moves ahead of a time step. Moreover, since all temporal operators in BOSL are bounded, the truth value of the formula is determined after a bounded number of time steps.

4.2. Syntax and semantics of BOSL

The logic BOSL has the same main temporal and logical operators of Linear Temporal Logic (LTL) [30]. However, as we described in the previous section, the models are different and the model checking technique we shall present in Section 4.4 is not standard.

Moreover, BOSL atomic formulae are propositions given in terms of equalities or inequalities of linear combinations of state variables and constants belonging to the following components of the state model: $\mathcal{D}, \mathcal{D}^{(*)}, \mathcal{C}$, and $\mathcal{P}$ (see Section 4.1).

With respect to other timed temporal logics that model check timed automata, such as MTL [31] or TCTL [20], the BOSL logic uses a completely different concept of model and is focused on properties based on a "collective emerging" behaviour of the system instead of classical properties in verification as safety or liveness properties.

The syntax of BOSL is as follows:

$$\begin{array}{lcl} \phi & ::= & \mathit{true} \mid p \mid \neg\phi \mid \phi \wedge \phi \mid \mathbf{X}\,\phi \mid \phi\,\mathbf{U}_{\prec m}\,\phi \mid D^{(h)}.\,\phi \\ p & ::= & \sum c_i v_i \sim b \end{array}$$

where $c_i, b \in \mathbb{R}$ are real numbers, $v_i \in \mathcal{D} \cup \mathcal{D}^{(*)} \cup \mathcal{P}$ are state variables, stored values or calculated state variables, $\sim \in \{<, \leq, >, \geq, =\}$, $\prec \in \{<, \leq\}$ and $m, h \in \mathbb{N}$. Note that we do not allow m to be 0 if the relation $<$ is used in a formula $\phi\,\mathbf{U}_{\prec m}\,\phi$.

states.

$$
\begin{array}{lcl}
\mathcal{M}, t \models \mathit{true} & \Leftrightarrow & \text{for all } \mathcal{M}, t \\
\mathcal{M}, t \models \sum_i c_i \cdot v_i \sim b & \Leftrightarrow & \sum_i c_i \cdot \mathcal{M}(v_i) \sim b \\
\mathcal{M}, t \models \neg\phi & \Leftrightarrow & \mathcal{M}, t \not\models \phi \\
\mathcal{M}, t \models \psi \wedge \phi & \Leftrightarrow & \mathcal{M}, t \models \psi \text{ and } \mathcal{M}, t \models \phi \\
\mathcal{M}, t \models \mathbf{X}\,\phi & \Leftrightarrow & \mathcal{M}, t+1 \models \phi \\
\mathcal{M}, t \models \psi\,\mathbf{U}_{\prec m}\,\phi & \Leftrightarrow & \begin{array}{l} \exists s_2 : 0 \leq s_2 \prec m \text{ such that } \mathcal{M}, t+s_2 \models \phi \\ \quad \text{and } \forall s_1 \in \{0, \ldots, s_2 - 1\} \;\; \mathcal{M}, t+s_1 \models \psi \end{array} \\
\mathcal{M}, t \models D^{(h)}.\phi & \Leftrightarrow & \mathcal{M}_{[\mathcal{D}^{(*)} := \mathcal{D}^{(*)} \cup \{d_1^h(t), \cdots, d_n^h(t)\}]}, t \models \phi
\end{array}
$$

Figure 3: Semantics of BOSL.

A BOSL formula has atomic propositions p, logical connectives $\neg$ $\wedge$, temporal operators $\mathbf{X}$ (next) $\mathbf{U}_{\prec m}$ (bounded until), and a *freeze* operator $D^{(h)}$ inspired to the freeze quantification of [32].

Informally, the formula $D^{(h)}.\,\phi$ stores in the current state $\mathcal{M}$, into the component $D^{(*)}$, the current values of the remaining times $d_i \in \mathcal{D}$ and then evaluates ϕ. The apex number h is intended as an identificator[5] of this particular freezing, so in the subformula ϕ the expression $d_i^{(h)}$ can be used to denote this freezed value of the remaining time of oscillator i. To have freezed values is useful in defining interesting synchronisation properties, as we show in Section 4.3.

Note also that BOSL uses a bounded version of the until operator $\mathbf{U}$ adding to it a constraint on the maximum time steps m to be considered.

As usual, we introduce shorthands by defining the following derivative logical and temporal operators:

$$
\begin{array}{lcll}
\psi \vee \phi & \Leftrightarrow & \neg(\neg\psi \wedge \neg\phi) & \text{or } (\vee) \\
\psi \rightarrow \phi & \Leftrightarrow & \neg\psi \vee \phi & \text{implies } (\rightarrow) \\
\psi \leftrightarrow \phi & \Leftrightarrow & (\psi \rightarrow \phi) \wedge (\phi \rightarrow \psi) & \text{equivalent } (\leftrightarrow) \\
\Diamond_{\prec m}\psi & \Leftrightarrow & \mathit{true}\,\mathbf{U}_{\prec m}\,\psi & \text{bounded eventually } (\Diamond) \\
\Box_{\prec m}\psi & \Leftrightarrow & \neg(\Diamond_{\prec m}\neg\psi) & \text{bounded always } (\Box)
\end{array}
$$

Figure 3 shows the semantics of the basic operators. Note that atomic propositions are checked directly using the values of variables $v_i \in \mathcal{D} \cup \mathcal{D}^{(*)} \cup$

[5]We do not allow the using of the same identification number h as identificator in two different occurrences of the freeze operator in the same formula.

$\mathcal{P}$ available in the state $\mathcal{M}$ and denoted $\mathcal{M}(v_i)$. Concerning the temporal operators, as usual $\mathbf{X}\,\phi$ is true if and only if ϕ is true at next step, and $\psi\,\mathbf{U}_{\prec m}\,\phi$ 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. Note that if $s_2 = 0$ the set of times in the $\forall$ is empty, meaning that the condition is trivially true. This corresponds to the case in which the subformula ϕ is immediately true at time t.

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

Proposition 4.1. *Let $\psi\,\mathbf{U}_{\prec m}\,\phi$ be a BOSL formula. Then, for all $\mathcal{M}, t$:*

$$\mathcal{M}, t \models \psi\,\mathbf{U}_{\leq m}\,\phi \;\Leftrightarrow\; \mathcal{M}, t \models \bigvee_{i=0}^{m} f_i(\psi, \phi)$$

and

$$\mathcal{M}, t \models \psi\,\mathbf{U}_{< m}\,\phi \;\Leftrightarrow\; \mathcal{M}, t \models \bigvee_{i=0}^{m-1} f_i(\psi, \phi)$$

where

$$f_i(\psi, \phi) = \begin{cases} \phi & \text{if } i = 0 \\ \psi \wedge \mathbf{X}\,(f_{i-1}(\psi, \phi)) & \text{if } i > 0 \end{cases}$$

Proof. The thesis is a well-known result of temporal logic applied in our case. It can be shown by induction on m applying the definition of the semantics. □

The previous result allows us to simplify the model checking algorithm, which thus deals only with the next temporal operator (see Section 4.4).

4.3. Examples of BOSL formulae

In this section we present some examples of collective synchronisation properties that can be expressed using the logic BOSL.

Example 4.2 (Synchronisation)**.** *Using the phase-coherence parameter r, it is possible to measure the collective behaviour of a system of smooth coupled oscillators. In particular Kuramoto [6] 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 synchronised, i.e. $r = 1$. The property that given a system of N oscillator timed automata with same*

frequencies, "it becomes perfectly synchronised within 10s", can be specified as:

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

Note that the subscript $\leq 10s$ of the diamond is not a natural number as requested in the syntax. We use this notation intending that the real subscript is $\leq m$ where $m = 10s/\delta t$, i.e. the number of time steps of length δt corresponding to 10 seconds.

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 synchronised with an approximation of ϵ and, after that, it remains synchronised for at least 5s":*

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

Example 4.3 (Locked and drifted oscillators)**.** *After a system of smooth coupled oscillators interacting with the Kuramoto model starts to synchronise, the population splits into a partially synchronised state consisting of two groups of oscillators: a synchronised group, called* locked *- that behaves with a frequency locked to the mean of the frequencies distribution - and a desynchronised 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 subset F of indexes of $A = \{i \mid 1 \leq i \leq N\}$ "the oscillators in F become eventually locked within 10s", can be expressed as:*

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

Note here the use of the freeze operator. $D^{(1)}$ stores the current values of the remaining times d_i in $\mathcal{D}^{(*)}$ with the identificator 1 and then, on the next time step, they are retrieved, using the notation $d_i^{(1)}$ with the identificator 1, to be compared with the new ones.

Example 4.4 (Pulse-coupled oscillators)**.** *Fifty years ago John Buck [1] provided a review of the synchronous rhythmic flashing of fireflies. The study of flash communication in many firefly species has revealed that timing relations*

between the flashes provide the necessary information in this system for sexual and species selection. In particular it has been shown that flash synchrony is pervasively, but enigmatically involved in the courtship. We can consider fireflies as a set of almost identical (same frequency) pulse-coupled oscillators, where the synchronisation happens when each firefly flashes. In this case we can consider a Peskin-like interaction semantics for the behaviour of the oscillator timed automata. Let $A = \{i \mid 1 \leq i \leq N\}$ *be the set of indexes of all oscillators representing* N *fireflies. The property that, given a firefly with index* z *"in the first 20 seconds, whenever* z *flashes all fireflies will be synchronised within 5s", can be expressed as:*

$$\phi^{A}_{\text{firefly}} = \Box_{\leq 20s}((d_z = 0) \rightarrow \Diamond_{\leq 5s} \, (\bigwedge_{i,j \in A} (d_i - d_j) = 0))$$

Note that, however, the first flashing of z *in the first 20 seconds is the interesting case. After that, if the formula is true, all fireflies are always synchronised, as the synchronisation is a persisting property over time in this synchronisation model.*

4.4. Model checking algorithm

Table 1 shows a high level description of the recursive model checking function `mcBOSL`. It takes a BOSL formula ϕ - where all the occurrences of the bounded until operator have been translated as shown in Proposition 4.1 - a state $\mathcal{M}$ (initially an initial state $\mathcal{M}_0$), a discretised time (initially 0), and a diagnostic 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 *addVariables* function allows to store the current state variables $d_i \in \mathcal{D}$ in the component $D^{(*)}$ of $\mathcal{M}$ whenever a freeze operator is met during the parsing.

For each $\mathbf{X}$ operator in the formula, a *nextStep* function is called in order to calculate the evolution of the system as described in Section 4.1. The function *nextStep* selects one of the (possibly different) successor states $\mathcal{M}'$ at time $t+1$ that can be generated due to the non-determinism of the progress relation step, as discussed in Section 4.1. Here we do not need to follow all the possible branches because they only change the "non-oscillating, internal" behaviours of the oscillator timed automata in the population. In other words, the BOSL formulae observe only the remaining times d_i in the state

function MCBOSL($\phi, \mathcal{M}, t$, dTrace)
▷ Returns (`true`, $\oslash$) or (`false`, a diagnostic trace)
select case
case $\phi = \mathit{true}$
return (`True`, $\oslash$)
end case
case $\mathit{isAtomic}(\phi)$
return $\mathit{checkAtomicProp}(\phi, \mathcal{M}, t, \text{dTrace})$
end case
case $\phi = \neg\phi_1$
return $\mathit{notDiag}(\text{mcBOSL}(\phi_1, \mathcal{M}, t, \text{dTrace}),\ \phi_1, \mathcal{M}, t, \text{dTrace})$
end case
case $\phi = \phi_1 \wedge \phi_2$
$\mathcal{M}' \leftarrow \mathcal{M}$
dTrace$' \leftarrow$ dTrace
return $\mathit{mergeAnd}($
mcBOSL$(\phi_1, \mathcal{M}', t, \text{dTrace}')$,
mcBOSL$(\phi_2, \mathcal{M}', t, \text{dTrace}'))$
end case
case $\phi = \mathbf{X}\,\phi_1$
dTrace $\leftarrow \mathit{append}(\text{dTrace}, \phi, \mathcal{M}, t)$
$\mathcal{M} \leftarrow \mathit{nextStep}(\mathcal{M})$
return mcBOSL $(\phi_1, \mathcal{M}, t+1, \text{dTrace})$
end case
case $\phi = D^{(h)}.\ \phi_1$
dTrace $\leftarrow \mathit{append}(\text{dTrace}, \phi, \mathcal{M}, t)$
$\mathcal{M} \leftarrow \mathit{addVariables}(\mathcal{M}, h)$
return mcBOSL $(\phi_1, \mathcal{M}, t, \text{dTrace})$
end case
end select
end function

Table 1: Model checking algorithm for BOSL.

model and the other values defined in $\mathcal{M}$. These times are calculated from the clocks $\overline{x}_i$ of the automata and, by the hypotheses of Definition 3.2, these are not affected by non-deterministic choices made on the underlying automata because we are assured that no matter which path is taken, every automaton T_i will fire its distinguished action after just $d_i = p_i - \nu_i(\overline{x}_i)$ time, where ν_i is the current clock valuation of automaton T_i.

Finally, the functions *notDiag* and *mergeAnd* simply manage the returning of a correct counterexample, given the results of the recursive calls.

The complexity of `mcBOSL` is determined in the following proposition.

Proposition 4.5. *Let m be the maximum depth of the syntactic tree of the formula ϕ and let N be the number of oscillators considered in $\mathcal{M}$. The complexity of the function* `mcBOSL`$(\phi, \mathcal{M}, t, \mathrm{dTrace})$ *in the worst case is* $O(2^m \cdot N)$.

Proof. We can estimate the time needed - in the worst case - to execute the function `mcBOSL` on a formula whose syntactic tree has depth m, as follows:

$$T(m) = \begin{cases} aN & \text{if} \quad m = 0 \\ 2T(m-1) + b & \text{if} \quad m > 0 \end{cases}$$

In the case $m = 0$ the algorithm checks the validity of an atomic proposition, i.e. the validity of a disequation with a number of variables proportional to the number of oscillators N. In the case $m > 0$ the worst situation is that of the $\wedge$ operator in which two recursive calls are to be done and a constant time b is needed to merge the results.

Unfolding the recursive definition of $T(m)$ we get $T(m) = 2T(m-1)+b = 4T(m-2)+2b+b = \ \{\text{after } k \text{ unfoldings}\} \ = 2^k T(m-k) + \sum_{i=0}^{k-1} b2^i$, where the last sum derives from the accumulation of the merging times. Thus, the complete unfolded definition is, when $k = m$, $T(m) = 2^m T(0) + b\sum_{i=0}^{m-1} 2^i = a2^m N + b(2^m - 1) = 2^m(aN + b) - b$ which is in $O(2^m \cdot N)$. □

A prototype implementation of the algorithm, managing only the case of Kuramoto interaction, can be found in the BOSL model checker, available at [33]. In order to overcome the high complexity of the algorithm we used some optimisations in the implementation, such as an iterative breadth-first search instead of a recursive depth-first search and efficient data structures to represent the state and to perform the requested checks on state variables. Moreover, to provide a good approximation of the relative time increment for

each oscillator timed automaton we needed, within each δt, to use a proper integration method that depends on the interaction function. In the case of Peskin, in which the interaction function is linear, the Euler method would have been sufficient, while the non-linear nature of the Kuramoto interaction function suggested the use of a Runge Kutta $4th$ order method [34]. The choice of the δt, as we mentioned before, is very important for the granularity of the analysis. In fact, we can not specify in a formula any interval of time that is not a multiple of δt. The performance of the resulted model checker is, in the average case, satisfying.

Let us show a simple analysis carried out with the model checker. We generated a system of 40 oscillators by choosing randomly the initial phases in the range $[0, 2\pi)$ and the frequencies following a CauchyLorentz distribution in the range $(0, 1)$. Figure 4 shows the distribution of the phases of each oscillator at the beginning. We set the coupling strength $k = 4.0$ and we check the formula $\Diamond_{\leq 10s}\ r > 0.98$. Figure 5 shows the result of the model checker that found this property to be true at 1.05s.

As a final remark we want to underline that the kind of analysis performed by the BOSL model checker could not be carried out by existing model checking tools for timed automata such as UPPAAL [25], KRONOS [23] or IF [35]. The main reasons are the completely different state space of the BOSL logic and the peculiarity of the successor function, depending on the interaction model, we use.

4.5. Case study: Pacemaker cells

The contractions of the heart are controlled by chemical impulses. The cells that create these rhythmical impulses are called pacemaker cells, and they directly control the heart rate. Pacemakers are also called the artificial devices that can be used after damage to the body's intrinsic conduction system to produce these impulses synthetically.

We can abstract the behaviour of a population of three pacemaker cells with the set of oscillator timed automata described in Figure 6 on the right. As the same figure shows on the left, after a phase of initial delay - less than the period of the oscillation - the cell fires an action potential, then undergoes a phase of depolarisation and finally it goes to the resting phase. The distinguish action of each oscillator timed automaton is **fire**. We chose the intrinsic pacemaker cycle length (T) of 485 ms, which is consistent with experimental data of a rabbit pacemaker cell published in the literature [36]. The aim of the case study is to verify some synchronisation properties of the

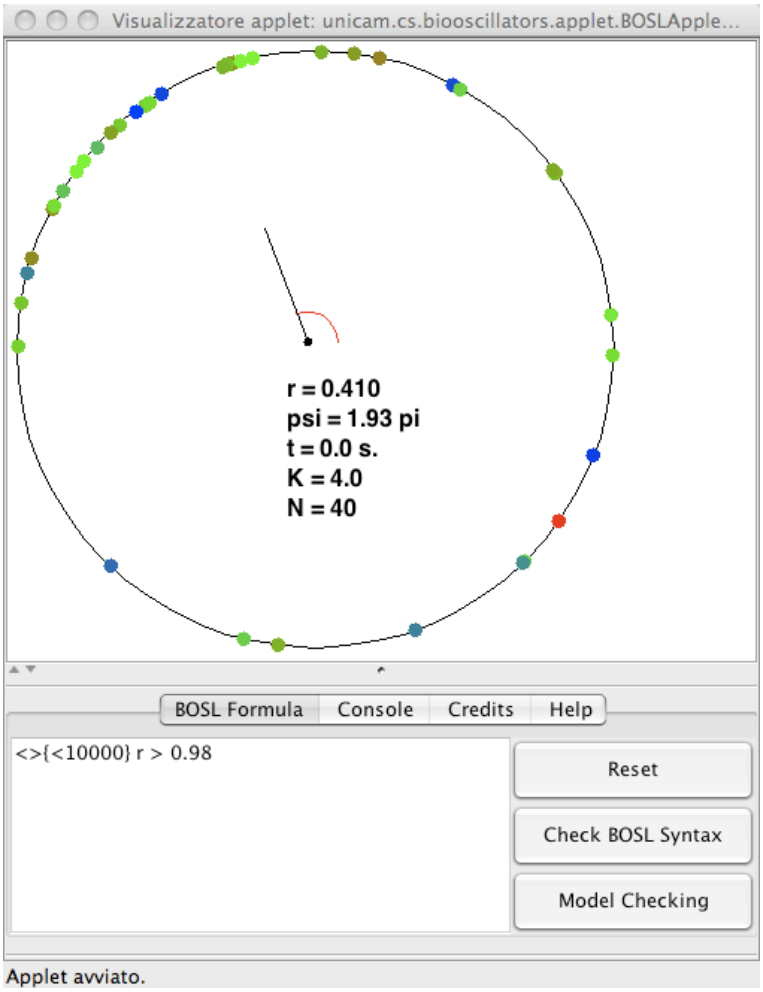

Figure 4: The initial configuration of the oscillators.

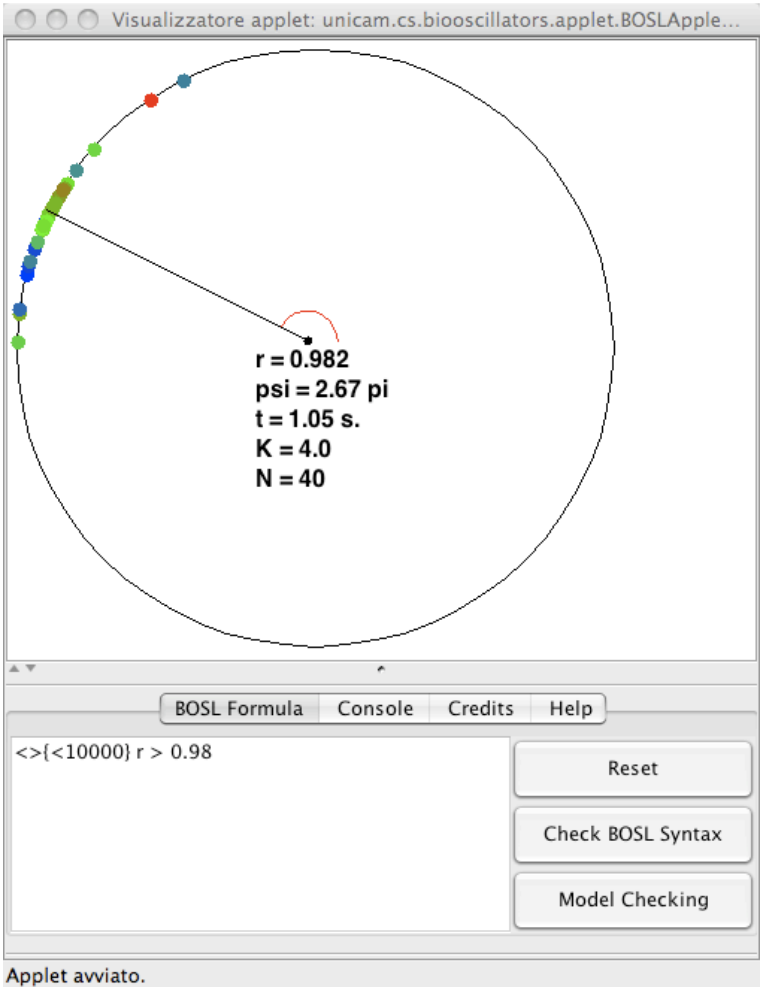

Figure 5: The verification of the formula $\Diamond_{\leq 10s}\, r > 0.98$.

population of oscillators shown in Figure 6. Since Peskin model [2] was used to study the pacemaker cells as pulsed-coupled oscillators, we used the Peskin interaction semantics discussed in Section 3.3.5. To find the parameters that generates an oscillator with a period of firing of $T = 485$ms, we chose $\gamma = 1$ and used the following formula to calculate S_0:

$$S_0 = \frac{e^{T\gamma}\gamma}{e^{T\gamma} - 1} = \frac{e^{0.485}}{e^{0.485} - 1} \approx 2.6$$

To calculate the corresponding value of phase of each oscillator i in the interval $[0 \dots 2\pi]$, given the remaining time t_d for the next distinguish action, we used the following formula:

$$p_i = \frac{-2\pi S_0}{\gamma} \left(\frac{1}{e^{\gamma(T - t_d)}} - 1 \right)$$

Figure 7 shows the behaviour of the oscillator timed automata without synchronisation ($\epsilon_t = 0$). Using the BOSL logic we can check if the synchronisation happens following a temporal sequence and within a certain amount of time. For example, we want to verify if, setting $\epsilon_t = 0.06$, "within 2 seconds, automaton A synchronises first with the automaton B and then, within maximum 2 natural pacemaker cycles, also with the automaton C":

$$\begin{aligned} \phi = \quad & \Diamond_{\leq 2s} \left(d_A = 0 \ \wedge d_B = 0 \ \wedge d_C \neq 0 \wedge \right. \\ & \left. \mathbf{X} \left(\Diamond_{\leq 0.97s} \left(d_A = 0 \ \wedge d_B = 0 \ \wedge d_C = 0 \right) \right) \right) \end{aligned}$$

As Figure 8 shows, this property is verified from the 917 ms. We can also check that automaton A never performs the distinguish action later than the automaton C using this formula:

$$\phi = \quad \Box_{\leq 2s} \left(d_A - d_C <= 0 \right)$$

The logic can be also used to find the right parameter ϵ_t such that a certain property is verified, by performing different simulations. This could be very beneficial in the future to control the oscillator interaction in order that a certain formula could be satisfied. An example could be finding the minimum ϵ_t (with an approximation of two digits after dot) such that within a maximum length of 3 natural pacemaker cycles we can obtain the full synchronisation:

$$\phi = \quad \Diamond_{\leq 1.455s} \left(d_A = 0 \ \wedge d_B = 0 \ \wedge d_C = 0 \right)$$

As Figures 8 and 9 show, the minimum ϵ_t, with two digits after dot such that this formula is true is 0.07.

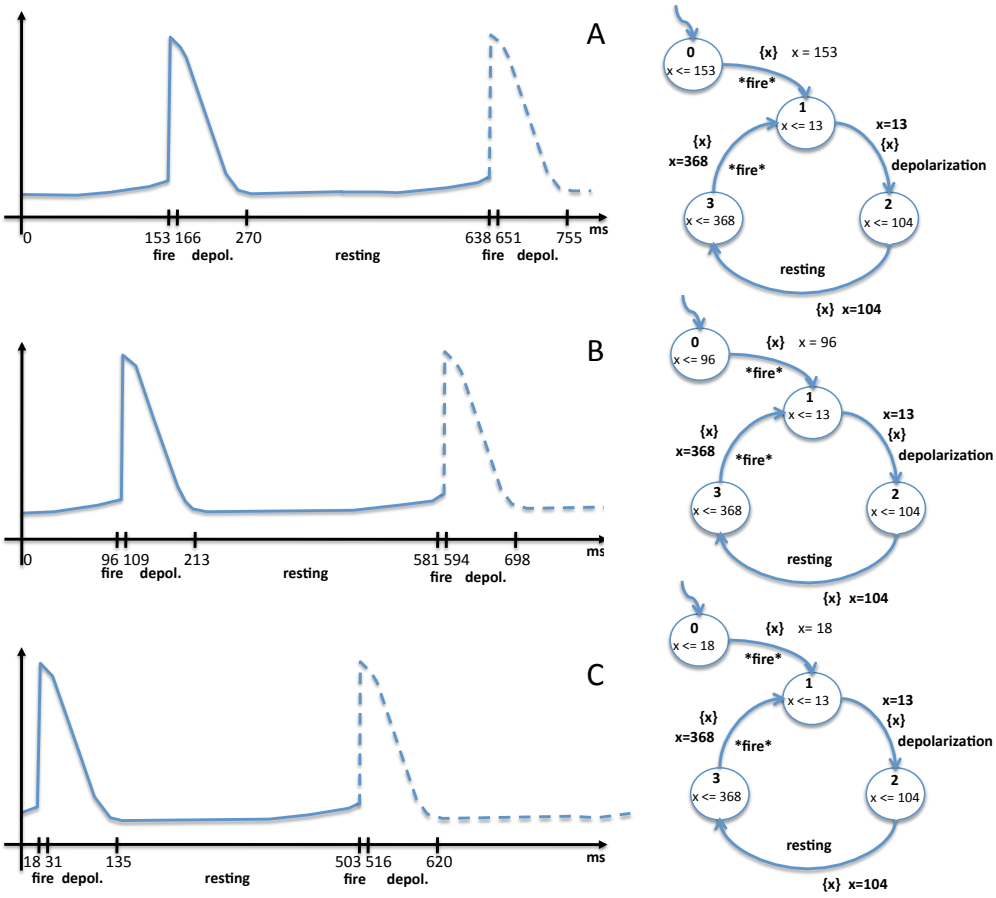

Figure 6: Oscillator timed automata for pacemaker cells.

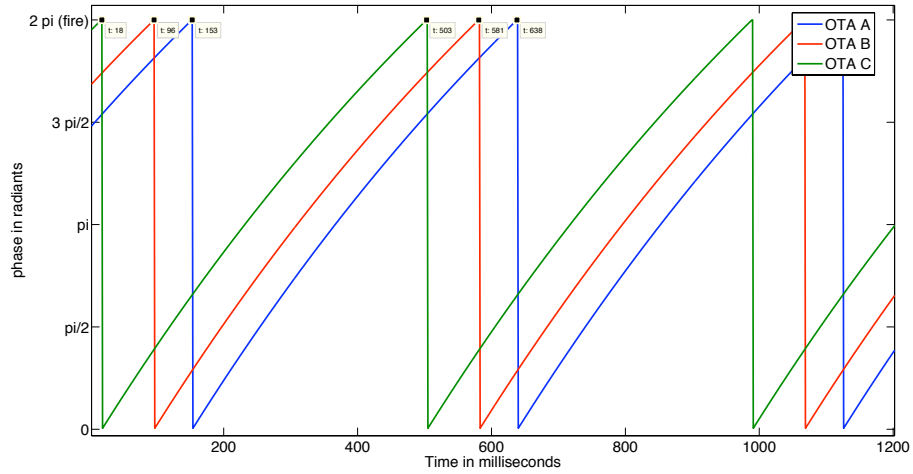

Figure 7: Oscillator timed automata without synchronisation.

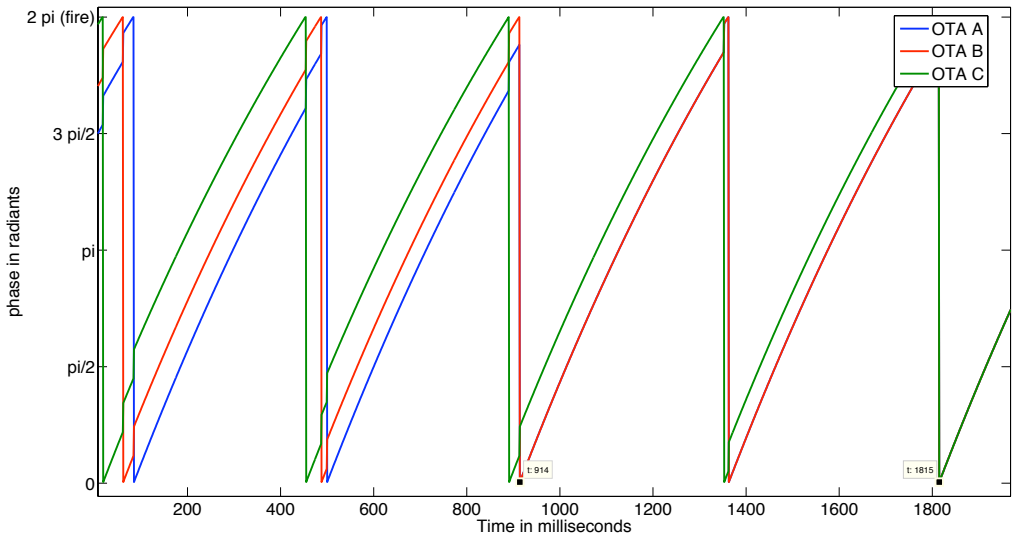

Figure 8: Self-synchronisation of oscillator timed automata with $\epsilon_t = 0.06$.

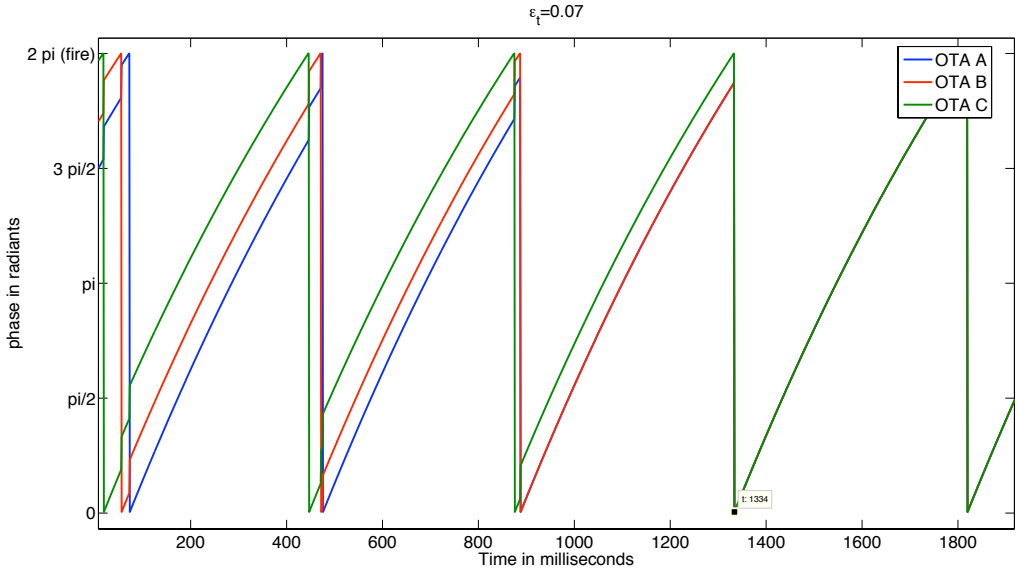

Figure 9: Self-synchronisation of oscillator timed automata with $\epsilon_t = 0.07$.

5. Conclusions

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 synchronisation, and we have instantiated it to the Kuramoto model, an example of smooth interaction, and to the Peskin model for pacemaker cells in the heart, an example of pulse interaction. A main advantage of the semantic definition is that it does not require changing the structure of the automata modelling the standalone oscillators. We have also introduced a logic, Biological Oscillator Synchronisation Logic (BOSL), in order to specify, and detect by model checking, collective synchronisation properties of a population of oscillator timed automata. A model checking algorithm has been presented and the implementation of a prototype of the model checker is available at [33]. Finally, a case study on pacemaker cells has been developed.

We remark that the objective of the modelling framework of oscillator timed automata we defined is not limited to the checking of BOSL properties, but it is intended to be a base for other analyses, such as checking of the internal states or internal events of some oscillators possibly in conjunction

with synchronisation properties, define control strategies depending on or triggered by the internal state of some oscillators, enlarge the synchronisation function in order to consider also the internal state of the oscillator, together with the relative distances of other oscillators in the population, to define the perturbation of time.

The main future directions of our work are: 1) extend BOSL logic with *control* operators that allow us to infer how to influence a set of oscillators by adding artificial oscillators - which we can control- or how to modify the parameters of some oscillators, in order to satisfy a desired synchronisation property. 2) Define a logic more expressive than BOSL in order to specify synchronisation-related properties depending also on the states and events of the oscillator timed automata considered, not only on the remaining times to the next firing. 3) A long term objective: adapt the framework and the model checking of the extended logic to control fibrillation in cardiac cells networks, exploiting other detection techniques already partially available (for more details of this line of research see [37]).

Acknowledgments

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

We wish to thank the anonymous reviewers for their constructive comments and suggestions.

[1] J. Buck, Synchronous rhythmic flashing of fireflies II, The Quarterly Review of Biology 63 (3) (1988) 265–289.

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

[3] J. Dye, Ionic and synaptic mechanisms underlying a brainstem oscillator: An in vitro study of the pacemaker nucleus of apteronotus, Journal of Comparative Physiology 168 (5) (1991) 521—-532.

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

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

[6] Y. Kuramoto, Phase dynamics of weakly unstable periodic structures: Condensed matter and statistical physics, Progress of theoretical physics 71 (6) (1984) 1182–1196.

[7] A. T. Winfree, Biological rhythms and the behavior of populations of coupled oscillators, Journal of Theoretical Biology 16 (1) (1967) 15 – 42. doi:10.1016/0022-5193(67)90051-3.

[8] R. Wang, L. Chen, K. Aihara, Synchronizing a multicellular system by external input: an artificial control strategy, Bioinformatics 22 (14) (2006) 1775–1781.

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

[10] C. S. Peskin, Mathematical Aspects of Heart Physiology, Courant Institute of Mathematical Sciences, 1975.

[11] E. Bartocci, F. Corradini, E. Merelli, L. Tesei, Model checking biological oscillators, Electronic Notes in Theoretical Computer Science 229 (1) (2009) 41 – 58, proceedings of the Second Workshop From Biology to Concurrency and Back (FBTC 2008). doi:10.1016/j.entcs.2009.02.004.

[12] Y. Kuramoto, Collective synchronization of pulse-coupled oscillators and excitable units, Physica D 50 (1) (1991) 15–30.

[13] R. E. Mirollo, S. H. Strogatz, Synchronization of pulse-coupled biological oscillators, SIAM Journal of Applied Mathematics 50 (6) (1990) 1645–1662.

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

[15] R. E. Mirollo, S. H. Strogatz, Synchronization of pulse-coupled biological oscillators, SIAM Journal of Applied Mathematics 50 (6) (1990) 1645–1662.

[16] L. F. Abbott, C. van Vreeswijk, Asynchronous states in networks of pulse-coupled oscillators, Phys. Rev. E 48 (2) (1993) 1483–1490.

[17] P. C. Bressloff, Mean-field theory of globally coupled integrate-and-fire neural oscillators with dynamic synapses, Phys. Rev. E 60 (2) (1999) 2160–2170.

[18] D. Golomb, D. Hansel, The number of synaptic inputs and the synchrony of large, sparse neuronal networks, Neural Comput 12 (2000) 1095–1139.

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

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

[21] V. Gupta, T. A. Henzinger, R. Jagadeesan, Robust timed automata, in: HART '97: Proceedings of the International Workshop on Hybrid and Real-Time Systems, Springer-Verlag, London, UK, 1997, pp. 331–345.

[22] D. N. Jansen, R. Wieringa, Extending ctl with actions and real time, J. Log. Comput. 12 (4) (2002) 607–621.

[23] C. Daws, A. Olivero, S. Tripakis, S. Yovine, The tool kronos, in: Proceedings of the DIMACS/SYCON workshop on Hybrid systems III : verification and control, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996, pp. 208–219.

[24] K. G. Larsen, P. Pettersson, W. Yi, Model-Checking for Real-Time Systems, in: Proc. of Fundamentals of Computation Theory, no. 965 in LNCS, 1995, pp. 62–88.

[25] G. Behrmann, A. David, K. G. Larsen, A tutorial on UPPAAL, in: Formal Methods for the Design of Real-Time Systems, SFM-RT 2004, no. 3185 in LNCS, Springer–Verlag, 2004, pp. 200–236.

[26] O. Maler, G. Batt, Approximating continuous systems by timed automata, in: FMSB, no. 5054 in LNCS, 2008, pp. 77–89.

[27] O. Stursberg, S. Kowalewski, S. Engell, On the generation of timed discrete approximations for continuous systems, Mathematical and Computer Modelling of Dynamical Systems 6 (1) (2000) 51–70.

[28] A. Olivero, J. Sifakis, S. Yovine, Using abstractions for the verification of linear hybrid systems, in: CAV, 1994, pp. 81–94.

[29] T. A. Henzinger, The theory of hybrid automata, in: LICS, 1996, pp. 278–292.

[30] A. Pnueli, The temporal logic of programs, in: FOCS, 1977, pp. 46–57.

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

[32] R. Alur, T. A. Henzinger, A really temporal logic, Journal of the ACM 41 (1) (2004) 181–203.

[33] E. Bartocci, F. Corradini, E. Merelli, L. Tesei, BOSL model checker prototype.
URL `http://cosy.cs.unicam.it/bosl/`

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

[35] M. Bozga, S. Graf, L. Mounier, If-2.0: A validation environment for component-based real-time systems, in: CAV, 2002, pp. 343–348.

[36] W. K. Bleeker, A. J. Mackaay, M. Masson-Pvet, L. N. Bouman, A. E. Becker, Functional and morphological organization of the rabbit sinus node, Circ. Res. 46 (1980) 11–22.

[37] E. Bartocci, F. Corradini, R. Grosu, E. Merelli, O. Riganelli, S. A. Smolka, Stonycam: A formal framework for modeling, analyzing and regulating cardiac myocytes, in: Concurrency, Graphs and Models: Essays Dedicated to Ugo Montanari on the Occasion of His 65th Birthday, no. 5065 in LNCS, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 493–502.