# Investigating Parametric Influence on Discrete Synchronisation Protocols using Quantitative Model Checking *

Paul Gainer, Sven Linker, Clare Dixon, Ullrich Hustadt, and Michael Fisher

University of Liverpool, UK
{p.gainer, s.linker, cldixon, u.hustadt, mfisher}@liverpool.ac.uk

**Abstract.** Synchronisation is an emergent phenomenon observable in nature. Natural synchronising systems have inspired the development of protocols for achieving coordination in a diverse range of distributed dynamic systems. Spontaneously synchronising systems can be mathematically modelled as coupled oscillators. In this paper we present a novel approach using model checking to reason about achieving synchrony for different models of synchronisation. We describe a general, formal population model where oscillators interact at discrete moments in time, and whose cycles are sequences of discrete states. Using the probabilistic model checker PRISM, we investigate the influence of various parameters of the model on the likelihood of, and time required for, achieving synchronisation.

## 1 Introduction

Synchronisation is an emergent phenomenon observable throughout nature; pacemaker cells in the sinoatrial node of the heart synchronise to set the rate and rhythm of a heartbeat, tree crickets chirp in harmony, and populations of fireflies synchronise their flashing to attract mates [6]. These decentralised natural systems have inspired the development of protocols for achieving synchrony in a diverse range of artificial decentralised systems; in particular swarm robotic systems and wireless sensor networks (WSNs). Applications include detecting faults in members of a robotic swarm [8], synchronising the duty cycles of sensor nodes in a network [19], auto-tuning mobile networks to save energy [4], and coordinating data dissemination for a WSN [5].

The cyclic behaviour of systems where synchrony spontaneously occurs can be modelled as networks of coupled oscillators with similar frequencies. Oscillators are *coupled* when some process results in the transferral of energy between them. In some systems oscillators are coupled such that their oscillations have a continuous influence on each other. The strength of the coupling between oscillators is determined by some global constant. When the mutual agitation

* This work was supported by both the Sir Joseph Rotblat Alumni Scholarship at Liverpool and the EPSRC Research Programme EP/N007565/1 *Science of Sensor System Software*.

of oscillators takes place only at discrete instances in time the oscillators are *pulse-coupled* [15,17]. At some distinguished point in the oscillation cycle a pulse-coupled oscillator *fires* and influences other nearby oscillators. An oscillator that is perturbed by another oscillator shifts or resets the phase of its own oscillation cycle to more closely match that of its neighbour. Over time this can lead to all oscillators matching phase, and synchronisation is achieved if all oscillators fire synchronously. In nature, the oscillation cycle of oscillators often includes a *refractory period*. The refractory period is an interval in the oscillation cycle during which its phase cannot be perturbed by other firing oscillators. This refractory period can prevent spurious mutual stimulation of the oscillators, which could lead to perpetual asynchrony. The introduction of a refractory period to oscillators in artificial systems not only helps to achieve synchrony, but can also be thought of as a period during which robots in a swarm, or nodes in a WSN, can turn off their wireless antennas and save energy.

The emergence of synchronisation in robot swarms, WSNs, and other distributed and dynamic systems is often investigated by designing and analysing simulations. Simulations can give detailed insight into how global behaviours of these systems emerge over time. Formal approaches can complement simulations, where desirable properties for the system can be unambiguously formulated, and rigorously checked against some formal model of the system, often finding corner cases that may not be covered even by a large number of tests of a simulation. Model checking has been successfully applied to qualitatively, and quantitatively, analyse control algorithms and protocols for both swarm systems [2,11,14] and WSNs [7,13]. In particular, model checking has been used to formally investigate the emergence of synchronisation in networks of oscillators. A general model of synchronisation for oscillators was introduced in [3], where oscillators were modelled as timed automata [1], and a model checking algorithm was used to determine the reachability of a synchronised state for distinguished runs of the model. More recently, Heidarian et al. [13] also used timed automata to exhaustively analyse a specific clock synchronisation protocol.

Our contribution in this paper is the development of a formal and general model for oscillator synchronisation, which is parameterised by a synchronisation model and a configuration for both oscillators and the network. In contrast to previous applications of model checking to detect synchronisation, our model is discrete. That is, the oscillators interact at discrete moments in time, and their oscillation cycles are defined as sequences of discrete states. Given an instantiation of our general model we automatically generate a discrete time Markov chain (DTMC). We discuss the results of model checking two instantiations of our model with regards to energy consumption.

In Sect. 2 we present the dynamics of individual oscillators with discrete oscillation cycles. We formally define our general, parameterised population model for a network of oscillators in Sect. 3, and describe how we construct the corresponding DTMC in Sect. 4. We discuss the results of checking synchronisation properties for two concrete instantiations of our formal model in Sect. 5. Concluding remarks and suggestions for further work are given in Sect. 6.

## 2 Discrete Oscillator Model

We consider a fully-connected network of $N$ pulse-coupled oscillators with identical dynamics over discrete time. We denote the set of these oscillators by $\mathcal{O} = \{1, \ldots, N\}$, where each $i \in \mathcal{O}$ corresponds to a single pulse-coupled oscillator. The *value* or *phase* of an oscillator $i$ in $\mathcal{O}$ at time $t$ is denoted by $\phi_i(t)$. The phase of an oscillator progresses through a sequence of discrete integer values bounded by some $T \geq 1$.

**Definition 1.** *The* evolution function *is a strictly increasing function* $\text{evol} : \{1, \ldots, T\} \to \mathbb{N}$ *with* $\text{evol}(\Phi) \geq \Phi$ *for all* $\Phi \in \{1, \ldots, T\}$*, that maps the current phase of an oscillator to its phase in the next discrete time step.*

We now introduce the *update function* and *firing predicate*, which respectively denote the updated phase of an oscillator $i$ at time $t$ after one time step, and the firing of oscillator $i$ at time $t$,

$$\mathit{update}_i(t) = \text{evol}(\phi_i(t)), \qquad \mathit{fire}_i(t) = \mathit{update}_i(t) > T.$$

The precise evolution of phase over time for an oscillator $i$ is then given by

$$\phi_i(t+1) = \begin{cases} 1 & \text{if } \mathit{fire}_i(t) \\ \mathit{update}_i(t) & \text{otherwise,} \end{cases}$$

where phase increases over time until $\text{evol}(\phi_i(t)) > T$, at which point oscillator $i$ *fires*, that is, $\phi_i(t+1)$ becomes 1 and the oscillator attempts to broadcast a firing signal to all other oscillators *coupled* to it. The phase progression of an uncoupled oscillator is cyclic, and we refer to one cycle as an *oscillation cycle*.

An oscillator's firing signal perturbs the phase of all coupled oscillators; we use $\alpha_i(t)$ to denote the number of all other oscillators in $\mathcal{O}$ that are coupled to $i$ and will broadcast their firing signal at time $t$. Furthermore, we define $\mu \in [0, 1]$ to be the probability that a *broadcast failure* occurs when an oscillator fires, that is, the attempt to broadcast its firing signal fails (the oscillator still resets its phase to 1). Note that $\mu$ is a global parameter, hence the chance of broadcast failure is identical for all oscillators. Observe that $\alpha_i(t)$ is defined globally even though the model is not deterministic, however we defer the reader to the detailed discussion of probabilities in the following section.

**Definition 2.** *The* perturbation function *is an increasing function* $\text{pert} : \{1, \ldots, T\} \times \mathbb{N} \times \mathbb{R}^+ \to \mathbb{N}$ *that maps the phase of an oscillator $i$, the number of oscillators that have fired and perturbed $i$, and a real value defining the strength of the coupling between oscillators, to an integer value corresponding to the induced perturbation to phase.*

We refine the update function to include the perturbation to phase induced by the firing of other oscillators that are *coupled* to oscillator $i$ at time $t$, $\mathit{update}_i(t) = \text{evol}(\phi_i(t)) + \text{pert}(\phi_i(t), \alpha_i(t), \epsilon)$. We can introduce a refractory period into the

oscillation cycle of each oscillator. A refractory period is an interval of discrete values $[1, R] \subseteq [1, T]$ where $0 \leq R \leq T$ is the size of the refractory period, such that if $\phi_i(t)$ is inside the interval, for some oscillator $i$ at time $t$, then $i$ cannot be perturbed by other oscillators to which it is coupled. If $R = 0$ then we set $[1, R] = \emptyset$, and there is no refractory period at all. To be consistent with the literature we only consider refractory periods that occur at the start of the oscillation cycle.

**Definition 3.** *The* refractory function $\text{ref} : \{1, \ldots, T\} \times \mathbb{N} \to \mathbb{N}$ *is defined as* $\text{ref}(\Phi, \Delta) = 0$ *if* $\Phi \in [1, R]$*, or* $\text{ref}(\Phi, \Delta) = \Delta$ *otherwise.*

Given $\Delta$, a degree of perturbance to the phase of an oscillator, and $\Phi$, the phase of that oscillator, $\text{ref}(\Phi, \Delta)$ returns 0 if $\Phi$ is in the refractory period defined by $R$, or $\Delta$ otherwise. We again amend the update function to include the refractory function, giving $update_i(t) = \text{evol}(\phi_i(t)) + \text{ref}(\phi_i(t), \text{pert}(\phi_i(t), \alpha_i(t), \epsilon))$.

## 3 Population Model

Let $\mathcal{O} = \{1, \ldots, N\}$ be a fully connected network of $N$ identical oscillators with phases in the range $1, \ldots, T$, whose dynamics are determined by the functions evol and pert, with a refractory period defined by $R$, coupled with strength $\epsilon \in [0, 1]$, and where the probability of broadcast failure is $\mu \in [0, 1]$. The *population model* of the network $\mathcal{O}$ is defined by $\mathcal{S} = (N, T, R, \epsilon, \text{evol}, \text{pert}, \mu)$.

Since all oscillators in our model are behaviourally identical we do not need to distinguish between oscillators sharing the same phase, and can reason about groups of oscillators, instead of individuals. The global state of the model is therefore a tuple, where each element $n_\Phi$ of the tuple $\langle n_1, \ldots, n_T \rangle$ corresponds to the number of oscillators sharing a phase value of $\Phi$. The population model does not account for the introduction of additional oscillators to a network, or the loss of existing coupled oscillators. That is, the population $N$ remains constant.

**Definition 4.** *A global state of a population model* $\mathcal{S} = (N, T, R, \epsilon, \text{evol}, \text{pert}, \mu)$ *is a* $T$*-tuple* $\pi \in \{0, \ldots, N\}^T$*, where* $\pi = \langle n_1, \ldots, n_T \rangle$ *and* $\sum_{\Phi=1}^{T} n_\Phi = N$*. The set of all global states of* $\mathcal{S}$ *is* $\Gamma(\mathcal{S})$*, or simply* $\Gamma$ *when* $\mathcal{S}$ *is clear from the context.*

*Example 1.* Figure 1 shows three global states for an instantiated population model, $\mathcal{S} = (5, 6, 2, 0.15, \text{evol}, \text{pert}, 0.1)$, where the synchronisation model described in [8] is instantiated by defining the evolution function as $\text{evol}(\Phi) = \Phi + 1$, and the perturbation function as $\text{pert}(\Phi, \alpha, \epsilon) = [\Phi \cdot \alpha \cdot \epsilon]$, where $[x]$ denotes $x$ rounded to the nearest integer. The label for each node $n_\Phi$ is the number of oscillators with phase $\Phi$. We omit the label if $n_\Phi = 0$. Oscillators at node $n_6$ are about to fire, and oscillators at nodes $n_1$ and $n_2$ are in their refractory period, and cannot be perturbed by the firing of other oscillators. The global states can be denoted by $\pi_0 = \langle 0, 1, 0, 2, 2, 0 \rangle$, $\pi_1 = \langle 0, 0, 1, 0, 2, 2 \rangle$, and $\pi_2 = \langle 5, 0, 0, 0, 0, 0 \rangle$. Later we will explain how transitions between these global states are made. Note that directional arrows indicate cyclic direction, and do not represent transitions.

With every global state $\pi$ we associate a non-empty set of *failure vectors*, where each failure vector is a tuple of broadcast failures that could occur in $\pi$.

**Definition 5.** *A* failure vector *is a $T$-tuple $B \in (\{0,\ldots,N\} \cup \{\star\})^T$. We denote the set of all possible failure vectors by $\mathcal{B}$.*

Given a failure vector $B = \langle b_1, \ldots, b_T \rangle$, $b_\Phi \in \{0,\ldots,N\}$ indicates the number of broadcast failures that occur for all oscillators with a phase of $\Phi$. If $b_\Phi = \star$ then no oscillators with a phase of $\Phi$ fire, for all $1 \leq \Phi \leq T$. Semantically, $b_\Phi = 0$ and $b_\Phi = \star$ differ in that the former indicates that all (if any) oscillators with phase $\Phi$ fire and no broadcast failures occur, while the latter indicates that all (if any) oscillators with a phase of $\Phi$ do not fire. If no oscillators fire at all in a global state then we have only one possible failure vector, namely $\{\star\}^T$.

### 3.1 Transitions

Later in this section we will describe how we can calculate the set of all possible failure vectors for a global state, and thereby identify all of its successor states. However we must first show how we can calculate the single successor state of a global state $\pi$, given some failure vector $B$.

*Absorptions.* For real deployments of synchronisation protocols it is often the case that the duration of a single oscillation cycle will be at least several seconds [8,18]. The perturbation induced by the firing of a group of oscillators may lead to groups of other oscillators to which they are coupled firing in turn. The firing of these other oscillators may then cause further oscillators to fire, and so forth, leading to a "chain reaction", where each group of oscillators triggered to fire is *absorbed* by the initial group of firing oscillators. Since the whole chain reaction of absorptions may occur within just a few milliseconds, and in our model the oscillation cycle is a sequence of discrete states, when a chain reaction occurs the phases of all perturbed oscillators are updated at one single time step.

Since we are considering a fully connected network of oscillators, two oscillators sharing the same phase will have their phase updated to the same value

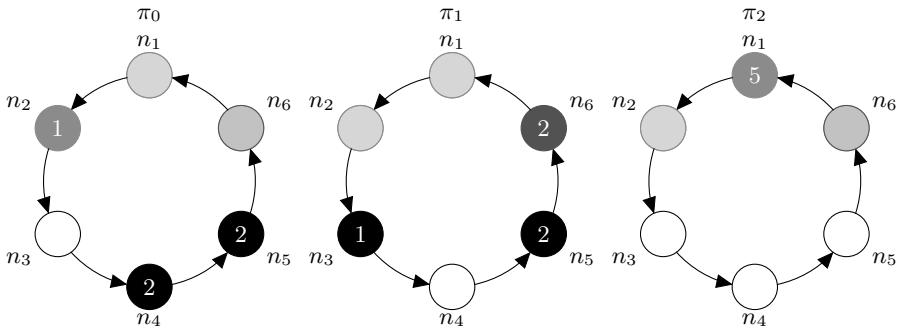



**Fig. 1.** Evolution of the global state over three discrete time steps.

in the next time step. They will always perceive the same number of other oscillators firing. Therefore, for each phase $\Phi$ we define the function $\alpha^{\Phi}\colon \Gamma \times \mathcal{B} \to \{1,\ldots,N\}$, where $\alpha^{\Phi}(\pi, B)$ is the number of oscillators with a phase greater than $\Phi$ perceived to be firing by oscillators with phase $\Phi$, in some global state, incorporating the broadcast failures defined in the failure vector $B$. This allows us to encode the aforementioned chain reactions of firing oscillators. Note that our encoding of chain reactions results in a global semantics that differs from typical parallelisation operations, for example, the construction of the crossproduct of the individual oscillators.

Given a global state $\pi = \langle n_, \ldots, n_T\rangle$ and a failure vector $B = \langle b_1, \ldots, b_T\rangle$, the following mutually recursive definitions show how we calculate the values $\alpha^1(\pi, B), \ldots, \alpha^T(\pi, B)$, and how functions introduced in the previous section are modified to indicate the update in phase, and firing, of *all* oscillators sharing the same phase $\Phi$. Observe that to calculate any $\alpha^{\Phi}(B, \pi)$ we only refer to definitions for phases greater than $\Phi$ and the base case is $\Phi = T$, that is, values are computed from $T$ down to 1.

$$\mathit{update}^{\Phi}(\pi, B) = \mathrm{evol}(\Phi) + \mathrm{ref}(\Phi, \mathrm{pert}(\Phi, \alpha^{\Phi}(\pi, B), \epsilon))$$

$$\mathit{fire}^{\Phi}(\pi, B) = \mathit{update}^{\Phi}(\pi, B) > T$$

$$\alpha^{\Phi}(\pi, B) = \begin{cases} 0 & \text{if } \Phi{=}T \\ \alpha^{\Phi+1}(\pi,B){+}n_{\Phi+1}{-}b_{\Phi+1} & \text{if } \Phi{<}T, b_{\Phi+1}{\neq}\star \text{ and } \mathit{fire}^{\Phi+1}(\pi,B) \\ \alpha^{\Phi+1}(\pi, B) & \text{otherwise} \end{cases}$$

*Transition Function.* We now define the transition function that maps phase values to their updated values in the next time step. Note that since we no longer differentiate between different oscillators with the same phase we only need to calculate a single value for their evolution and perturbation.

**Definition 6.** *The* phase transition function $\tau : \Gamma \times \{1,\ldots,T\} \times \mathcal{B} \to \mathbb{N}$ *maps a global state, a phase $\Phi$, and some possible failure vector $B$ for $\pi$, to the updated phase in the next discrete time step, with respect to the broadcast failures defined in $B$, and is defined as*

$$\tau(\pi, \Phi, B) = \begin{cases} 1 & \mathit{if}\ \mathit{fire}^{\Phi}(\pi, B) \\ \mathit{update}^{\Phi}(\pi, B) & \mathit{otherwise.} \end{cases}$$

**Lemma 1.** *The range of the function $\tau$ is bound by $T$. That is, for any $\pi$, for any possible failure vector $B$ for $\pi$, and for all $\Phi \in \{1,\ldots,T\}$, we have that $1 \leq \tau(\pi, \Phi, B) \leq T$.*

*Proof.* By construction.

Let $\mathcal{U}_{\Phi}(\pi, B)$ be the set of phase values $\Psi$ where all oscillators with phase $\Psi$ in $\pi$ will have their phase updated to $\Phi$ in the next time step, with respect to the broadcast failures defined in $B$. Formally,

$$\mathcal{U}_{\Phi}(\pi, B) = \{\Psi \mid \Psi \in \{1,\ldots,T\} \wedge \tau(\pi, \Psi, B) = \Phi\}.$$

We can now calculate the successor state of a global state $\pi$ and define how the model evolves over time.

**Definition 7.** *The* successor function $\text{succ} : \Gamma \times \mathcal{B} \to \Gamma$ *maps a global state* $\pi$ *and a failure vector* $B$ *to a state* $\pi'$*, and is defined as* $\text{succ}(\langle n_1, \ldots, n_T \rangle, B) = \langle n'_1, \ldots, n'_T \rangle$*, where* $n'_\Phi = \sum_{\Psi \in \mathcal{U}_\Phi(\pi, B)} n_\Psi$ *for* $1 \leq \Phi \leq T$*.*

*Example 2.* Consider the global state $\pi_0$ of Fig 1 where no oscillators will fire since $n_6 = 0$. We therefore have one possible failure vector for $\pi_0$, namely $B = \{\star\}^6$. Since no oscillators fire the dynamics of the oscillators are determined solely by evol, and all oscillators simply increase their phase by 1 in the next time step. Now consider the global state $\pi_1$ and $B = \langle \star, \star, 0, 0, 0, 1 \rangle$, a possible failure vector for $\pi_1$, indicating that oscillators with phases of 3 to 6 will fire and one broadcast failure will occur for one of the two oscillators that will fire with phase 6. Despite the broadcast failure occurring, a chain reaction will occur as the firing of the single oscillator with phase 6 will perturb the two oscillators with phase 5 to fire also. The combined perturbation induced by the firing of all three oscillators will cause the final oscillator with phase 3 to fire. All oscillators are therefore absorbed into the initial group of firing oscillators. Since $\mathit{fire}^6(\pi_1, B)$ holds we have that $\alpha^5(\pi_1, B) = \alpha^6(\pi_1, B) + n_6 - b_6 = 0 + 2 - 1 = 1$. Similarly since $\mathit{fire}^5(\pi_1)$ holds we have that $\alpha^4(\pi_1, B) = \alpha^5(\pi_1, B) + n_5 = 1 + 2 = 3$. We then continue calculating $\alpha^\Phi(\pi_1, B)$ for $3 \geq \Phi \geq 1$, and conclude that $\mathcal{U}_1(\pi_1, B) = \{6, 5, 4, 3\}$ and $\mathcal{U}_6(\pi_1, B) = \mathcal{U}_5(\pi_1, B) = \mathcal{U}_4(\pi_1, B) = \emptyset$. Since $R = 2$ we have that $\mathcal{U}_3(\pi_1, B) = \{2\}$ and $\mathcal{U}_2(\pi_1, B) = \{1\}$. We calculate the successor of $\pi_1$ as $\pi_2 = \text{succ}(\langle 0, 0, 1, 0, 2, 2 \rangle, B) = \langle n_6 + n_5 + n_4 + n_3, n_1, n_2, 0, 0, 0 \rangle = \langle 5, 0, 0, 0, 0, 0 \rangle$.

**Lemma 2.** *The number of oscillators is invariant during transitions, i.e., the successor function only creates tuples that are states of the given model. Formally, let* $\pi = \langle n_1, \ldots, n_T \rangle$ *and* $\pi' = \langle n'_1, \ldots, n'_T \rangle$ *be two states of a model* $\mathcal{S}$ *such that* $\pi' = \text{succ}(\pi, B)$*, where* $B$ *is some possible failure vector for* $\pi$*. Then* $\sum_{\Phi=1}^{T} n_\Phi = \sum_{\Phi=1}^{T} n'_\Phi = N$*.*

*Proof.* By construction.

### 3.2 Failure Vector Calculation

We construct all possible failure vectors for a global state by considering every group of oscillators in decreasing order of phase. At each stage we determine if the oscillators would fire. If they fire then we consider each outcome where any, all, or none of the firings result in a broadcast failure. We then add a corresponding value to a partially calculated failure vector and consider the next group of oscillators with a lower phase. If the oscillators do not fire then there is nothing left to do, since by Def. 1 we know that evol is strictly increasing, and by Def. 2 we know that pert is increasing, therefore all oscillators with a lower phase will also not fire. We can then pad the partial failure vector with $\star$ appropriately to indicate that no failure could happen since no oscillators fired.

Table 1 illustrates how a possible failure vector for global state $\pi_1$ in Fig. 1 is iteratively constructed. The first three columns respectively indicate the current iteration $i$, the global state $\pi_1$ with the currently considered oscillators indicated, and the elements of the failure vector $B$ computed so far. The fourth column is *true* if the oscillators with phase $T{+}1{-}i$ would fire given the broadcast failures in the partial failure vector. We must consider all outcomes of any or all firings resulting in broadcast failure. The final column therefore indicates whether the value added to the partial failure vector in the current iteration is the only possible value (*false*), or if a choice from one of several possible values (*true*).

Initially we have an empty partial failure vector. At the first iteration there are 2 oscillators with a phase of 6. These oscillators will fire so we must consider each case where $0, 1$ or 2 broadcast failures occur. Here we choose 1 broadcast failure, which is then added to the partial failure vector. At iteration 2, oscillators with a phase of 5 fire, and again we must consider each case with $0, 1$ or 2 broadcast failures occur; here we choose 0. At iteration 3 oscillators with a phase of 4 would have fired, but since there are no oscillators with a phase of 4 we only have one possible value to add to the partial failure vector, namely 0. At iteration 4 a single oscillator with a phase of 3 fires, and we choose the case where the firing did not result in a broadcast failure. In the final iteration oscillators with a phase of 2 do not fire, hence we can conclude that oscillators with a phase of 1 also do not fire, and can pad the partial failure vector appropriately with $\star$.

Formally, we define a family of functions $f$ indexed by $\Phi$, where each $f_\Phi$ takes as parameters some global state $\pi$, and $V$, a vector of length $T-\Phi$. $V$ represents all broadcast failures for all oscillators with a phase greater than $\Phi$. The function $f_\Phi$ then computes the set of all possible failure vectors for $\pi$ with suffix $V$. Here we use the notation $v^\frown v'$ to indicate vector concatenation.

**Definition 8.** *We define* $f_\Phi : \Gamma \times \{0, \ldots, N\}^{T-\Phi} \to \mathbb{P}((\{0, \ldots, N\} \cup \{\star\})^T)$*, for* $1 \leq \Phi \leq T$*, as the family of functions indexed by* $\Phi$*, where* $\pi = \langle n_1, \ldots, n_T \rangle$ *and*

$$f_\Phi(\pi, V) = \begin{cases} \bigcup_{k=0}^{n_\Phi} f_{\Phi-1}(\pi, \langle k \rangle^\frown V) & \text{if } 1 < \Phi \leq T \text{ and } \mathit{fire}^\Phi(\pi, \{\star\}^{\Phi\frown} V) \\ \bigcup_{k=0}^{n_1} \{\langle k \rangle^\frown V\} & \text{if } \Phi = 1 \text{ and } \mathit{fire}^1(\pi, \langle \star \rangle^\frown V) \\ \{\{\star\}^{\Phi\frown} V\} & \text{otherwise} \end{cases}$$

**Definition 9.** *Given a global state* $\pi \in \Gamma$*, we define* $\mathcal{B}_\pi$*, the set of all possible failure vectors for that state, as* $\mathcal{B}_\pi = f_T(\pi, \langle\rangle)$*, and define* $\mathit{next}(\pi)$*, the set of all successor states of* $\pi$*, as* $\mathit{next}(\pi) = \{\mathrm{succ}(\pi, B) \mid B \in \mathcal{B}_\pi\}$.

**Table 1.** Construction of a possible failure vector for a global state $\pi_1 = \langle 0, 0, 1, 0, 2, 2 \rangle$.

| iteration ($i$) | $\pi_1$ | failure vector $B$ | fired | branches |
|---|---|---|---|---|
| 0 | $\langle 0, 0, 1, 0, 2, 2 \rangle$ | $\langle\rangle$ | – | *false* |
| 1 | $\langle 0, 0, 1, 0, 2, \underline{2} \rangle$ | $\langle 1 \rangle$ | *true* | *true* |
| 2 | $\langle 0, 0, 1, 0, \underline{2}, 2 \rangle$ | $\langle 0, 1 \rangle$ | *true* | *true* |
| 3 | $\langle 0, 0, 1, \underline{0}, 2, 2 \rangle$ | $\langle 0, 0, 1 \rangle$ | *true* | *false* |
| 4 | $\langle 0, 0, \underline{1}, 0, 2, 2 \rangle$ | $\langle 0, 0, 0, 1 \rangle$ | *true* | *true* |
| 5 | $\langle 0, \underline{0}, 1, 0, 2, 2 \rangle$ | $\langle \star, \star, 0, 0, 0, 1 \rangle$ | *false* | – |

Note that for some global states $|next(\pi)| < |\mathcal{B}_\pi|$, since we may have that $\text{succ}(\pi, B) = \text{succ}(\pi, B')$ for some $B, B' \in \mathcal{B}_\pi$ with $B \neq B'$.

Given a global state $\pi$ and a failure vector $B \in \mathcal{B}_\pi$, we will now compute the probability of a transition being made to state $\text{succ}(\pi, B)$ in the next time step. Recall that $\mu$ is the probability with which a broadcast failure occurs. Firstly we define the probability mass function $\mathrm{P}_{fail} : \{1, \ldots, N\} \times \{1, \ldots, N\} \to [0, 1]$, where $\mathrm{P}_{fail}(n, b)$ gives the probability of $b$ broadcast failures occurring given that $n$ oscillators fire, as $\mathrm{P}_{fail}(n, b) = \mu^b(1-\mu)^{n-b}\binom{n}{b}$. We then denote by $\mathrm{P}_\tau(\pi) : \mathcal{B}_\pi \to [0, 1]$ the function mapping a possible broadcast failure vector $B$ for $\pi$, to the probability of the failures in $B$ occurring. That is,

$$\mathrm{P}_\tau(\langle n_1, \ldots, n_T \rangle)(\langle b_1, \ldots, b_T \rangle) = \prod_{\Phi=1}^{T} \begin{cases} \mathrm{P}_{fail}(n_\Phi, b_\Phi) & \text{if } b_\Phi \neq \star \\ 1 & \text{otherwise} \end{cases}$$

**Lemma 3.** *For any global state $\pi$, $\mathrm{P}_\tau(\pi)$ is a discrete probability distribution over $\mathcal{B}_\pi$. Formally, $\sum_{B \in \mathcal{B}_\pi} \mathrm{P}_\tau(\pi)(B) = 1$.*

*Proof.* By induction over a tree where internal nodes are partially constructed failure vectors and leaf nodes are failure vectors.

*Example 3.* We consider again the global states $\pi_1 = \langle 0, 0, 1, 0, 2, 2 \rangle$ and $\pi_2 = \langle 5, 0, 0, 0, 0, 0 \rangle$, given in Fig. 1, of the population model instantiated in Example 1, and the failure vector $B = \langle \star, \star, 0, 0, 0, 1 \rangle$ given in Example 2, noting that $B \in \mathcal{B}_{\pi_1}$, $\text{succ}(\pi_1, B) = \pi_2$, and $\mu = 0.1$. We calculate the probability of a transition being made from $\pi_1$ to $\pi_2$ as $\mathrm{P}_\tau(\langle 0, 0, 1, 0, 2, 2 \rangle)(\langle \star, \star, 0, 0, 0, 1 \rangle) = 1 \cdot 1 \cdot \mathrm{P}_{fail}(1, 0) \cdot \mathrm{P}_{fail}(0, 0) \cdot \mathrm{P}_{fail}(2, 0) \cdot \mathrm{P}_{fail}(2, 1) = 1 \cdot 1 \cdot 0.9 \cdot 1 \cdot 0.81 \cdot 0.18 = 0.13122$.

We now have everything we need to fully describe the evolution of the global state of a population model over time. A *run* of a population model $\mathcal{S}$ is an infinite sequence of global states $\Pi = \pi_0 \to \pi_1 \to \pi_2 \to \pi_3 \cdots$, where $\pi_0$ is called the *initial state*, and for all $k \geq 0$, $\pi_k \to \pi_{k+1}$ if, and only if, $\pi_{k+1} \in next(\pi)$. We denote the set of all possible runs of $\mathcal{S}$ by $\Pi(\mathcal{S})$.

### 3.3 Synchronisation

When all oscillators in a population model have the same phase in a global state we say that the state is *synchronised*. Formally, a global state $\pi = \langle n_1, \ldots, n_T \rangle$ is *synchronised* if, and only if, there is some $\Phi \in \{1, \ldots, T\}$ such that $n_\Phi = N$. Hence, for all $\Phi' \neq \Phi$, we have that $n_{\Phi'} = 0$. We use the notation $synch(\pi)$ to indicate that some global state $\pi$ is synchronised. We will often want to reason about whether some particular run $\Pi$ of a model leads to a global state that is synchronised. We say that a run $\Pi = \pi_0 \to \pi_1 \to \ldots$ *synchronises* if, and only if, there exists some $k \geq 0$ with $synch(\pi_k)$. We use the notation $synch(\Pi)$ to indicate that some run $\Pi$ synchronises. Once a synchronised global state is reached any successor states will remain synchronised. Finally we can say that a model *synchronises* if, and only if, $synch(\Pi)$ for all $\Pi \in \Pi(\mathcal{S})$. In Fig. 1 global state $\pi_2$ is synchronised, since $n_1 = N$.

## 4 Model Generation

We choose to use the probabilistic model checker PRISM [16] to formally verify properties of our model. Given a probabilistic model of a system, PRISM can be used to reason about both temporal and probabilistic properties of the input model, by checking some requirement expressed in a suitable formalism against all possible runs of the model. We define our input models as *Discrete Time Markov Chains* (DTMCs). A DTMC is a tuple $(Q, \mathit{init}, \mathrm{P})$ where $Q$ is a set of states, $\mathit{init} \in Q$ is the initial state, and $\mathrm{P} : Q \times Q \to [0, 1]$ is the function mapping ordered pairs of states $(q, q')$ to the probability with which a transition from $q$ to $q'$ occurs, where $\sum_{q' \in Q} \mathrm{P}(q, q') = 1$ for all $q \in Q$.

Given a population model $\mathcal{S} = (N, T, R, \epsilon, \mathrm{evol}, \mathrm{pert}, \mu)$, we construct a DTMC $(Q, \mathit{init}, \mathrm{P})$. We define the set of states $Q$ to be $\Gamma(\mathcal{S}) \cup \{\mathit{init}\}$. In the initial state all oscillators are considered to be *unconfigured*. That is, oscillators have not yet been assigned a value for their phase. For each $q \in Q$, where $q \in \Gamma(\mathcal{S})$ and $q = \langle n_1, \ldots, n_t \rangle$, we define

$$\mathrm{P}(\mathit{init}, q) = \frac{1}{T^N} \cdot \prod_{i=1}^{T} \binom{N - (\sum_{j=1}^{i-1} n_j)}{n_i}$$

to be the probability of moving from *init* to a state where the oscillators are *configured* with the phase values defined in $q$, since there are $N$ choose $n_1$ ways to select $n_1$ oscillators to have a phase of 1, then $N - n_1$ choose $n_2$ ways to select $n_2$ oscillators to have a phase of 2, and so forth. For every $q \in Q \setminus \mathit{init}$ we consider each $q' \in Q \setminus \mathit{init}$ where $q' = \mathrm{succ}(q, B)$ for some $B \in \mathcal{B}_q$, and set $\mathrm{P}(q, q') = \mathrm{P}_\tau(q)(B)$. For all other $q \in Q \setminus \mathit{init}$ and $q' \in Q$, where $q \neq q'$ and $q' \notin \mathit{next}(q)$, we set $\mathrm{P}(q, q') = 0$.

A state in PRISM is a valuation for a set of variables over the domain consisting of finitely bound booleans and integers. Global states in $\Gamma$ are encoded using $T$ finitely bound integer variables ranging over $N$ discrete values. To facilitate the analysis of many different oscillator population models we provide a Python script[1] that allows the user to define ranges for $N$, $T$, $R$, $\epsilon$ and $\mu$, for some fixed definitions for evol and pert. Then, given a list of properties, for each combination of parameters the script generates a model, checks all properties using PRISM, and writes user specified output (e.g. result, model checking time, etc.) to a comma separated value file which can be used by statistical analysis tools. Even though models in PRISM can be parametric, the parameters may only be used to describe probabilities. Therefore, our generated models can only be parameterised by $\mu$, since changing the value of $\mu$ does not result in the loss or addition of any transitions to the model.

[1] The model generation script and the results presented in this paper can be found at `https://github.com/PaulGainer/mc-bio-synch`

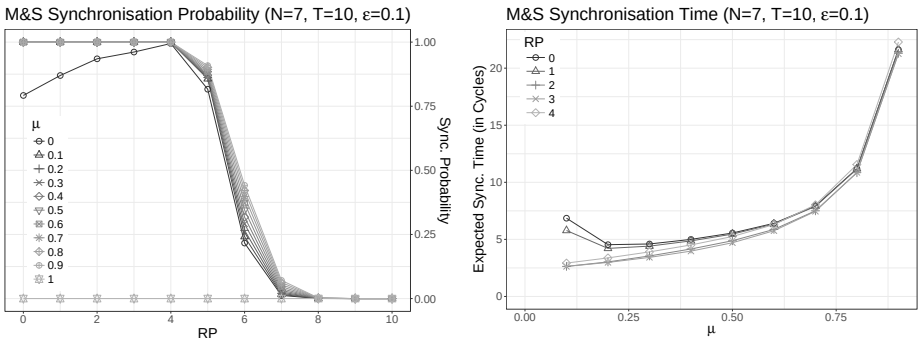
## 5 Evaluation

Within this section we will discuss the properties of two instantiations of the formal model defined in Sect. 3. To that end, we created concrete models for PRISM for different parameters of the models, e.g. number of oscillators and different coupling constants. Each of these models was subsequently checked by PRISM with respect to different properties. Other case studies could also be considered for alternative models of synchronisation where the dynamics of oscillators, and their interactions, can be described by some evolution and perturbation function.

Properties to be checked are specified using *Probabilistic Computation Tree Logic* (PCTL) [12]. PCTL consists of classical logical operators, temporal operators including $\Diamond\varphi$, "at some future point $\varphi$ holds", and the probabilistic operator $\mathbf{P}_{\bowtie\gamma}[\varphi]$, where $\bowtie$ is a relational operator and $\gamma \in [0,1]$ is a probability threshold. We can therefore specify properties such as $\mathbf{P}_{\geq 0.1}[\Diamond\varphi]$, "$\varphi$ holds at some future point with a probability of at least 0.1". In addition to assertions, PRISM allows the specification of properties that evaluate to a numerical value, using the syntax $\mathbf{P}_{=?}[\varphi]$, "what is the probability that $\varphi$ holds?". Furthermore, rewards can be associated with states, and the reachability reward operator $\mathbf{R}$ can be used to calculate expected rewards. For example $\mathbf{R}_{=?}[\Diamond\varphi]$ expresses "what is the expected reward for reaching a state where $\varphi$ holds?".

We are interested in the probability of eventual synchronisation and in the average time needed to achieve synchronisation. We formalise these properties in PCTL as $\varphi_1 = \mathbf{P}_{=?}[\Diamond\ \mathit{synchronised}]$, and $\varphi_2 = \mathbf{R}_{\{\mathit{time\_to\_synch}\}=?}[\Diamond\ \mathit{synchronised}]$. In these formulas *synchronised* is a name for the formula $\bigvee_{i=1}^{T} n_i = N$ used within the PRISM model, while *time_to_synch* is a reward structure associating a value of $\frac{1}{T}$ with each state where oscillators are configured (i.e., not *init*) and where the system is not synchronised, that records the number of cycles taken to achieve synchrony. As a consequence, the result of model checking with respect to $\varphi_2$ is the expected value of the reward *time_to_synch* accumulated along a path until synchrony occurs. Observe that PRISM gives a result of *Infinity* for accumulating *time_to_synch* along a run that does not synchronise. Since PRISM computes the expected value over all paths, this implies that a system with non-synchronising paths will also result in *Infinity* for $\varphi_2$.

In the following, we present the model checking results for two instantiations of our model. We will discuss these results and the resulting trade-offs for parameter choices. For a network of sensor nodes, several attributes can be weighted against each other: (i) probability of synchronisation, (ii) time for achieving synchrony, (iii) battery life of a single oscillator. While we get direct results for the first two properties from PRISM, battery life is dependent on the energy consumption, which can only be estimated from the parameters of the model. In WSNs, communication is costly with respect to energy consumption. Communication is either active when sending a message, i.e., when a node fires, or passive, when receiving messages from other nodes. Hence, during periods where a sensor does neither, the antenna can be shut down to save energy. In our models, this interval of inactivity corresponds to the refractory period. That is, the longer the refractory period is, the less energy will be consumed.

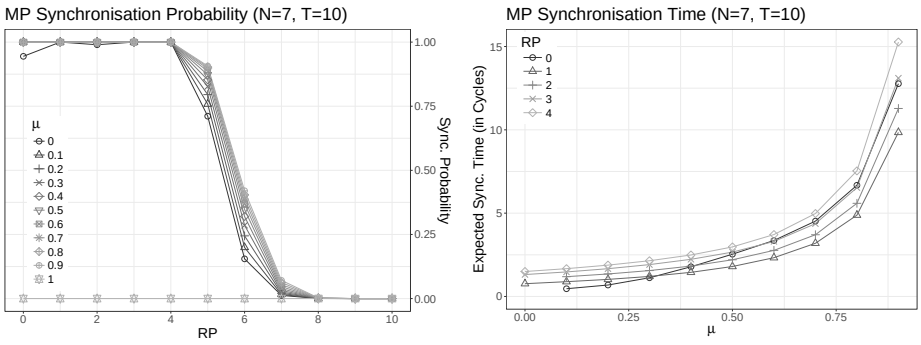



(a) (b)

**Fig. 2.** Mirollo & Strogatz synchronisation: synchronisation probabilities for different refractory periods, and synchronisation times for different rates of broadcast failure.

*Mirollo and Strogatz Synchronisation Model.* Here, we present the results of model checking population models where the perturbation function is a discretisation of the Mirollo and Strogatz (M&S) model of synchronisation used by Perez et al. [18], namely $\text{pert}(\Phi, \alpha, \epsilon) = [\Phi \cdot \alpha \cdot \epsilon]$. Note that, here, the perturbation induced by the firing of another oscillator increases linearly with the phase of the perturbed oscillator. The evolution function is simply the successor function, $\text{evol}(\Phi) = \Phi + 1$. With these functions fixed, we created models for different numbers of oscillators $3 \leq N \leq 7$, cycle lengths $4 \leq T \leq 10$, coupling constants $\epsilon \in \{0, 0.1, \ldots, 1.0\}$, refractory periods $0 \leq R \leq T$, and probabilities of message loss $\mu \in \{0, 0.1, \ldots, 1.0\}$. We used PRISM to analyse models with respect to $\varphi_1$ and $\varphi_2$.

Figure 2a plots the probability of synchronisation for different rates of broadcast failure against the refractory period for $N{=}7$, $T{=}10$, and $\epsilon{=}0.1$. We can extrapolate a trade-off between a high refractory period and high synchronisation probability. As long as the refractory period is less than half the oscillation cycle, synchronisation will be achieved in almost all cases. Higher values for $R$ result in a rapid drop in synchronisation probability. The exceptions are the edge cases $\mu{=}0$ and $\mu{=}1$. Unsurprisingly, if all firings result in broadcast failures ($\mu{=}1$), the synchronisation probability is almost zero. In fact, the only runs that synchronise in this case are runs whose initial states are synchronised. The comparably bad synchronisation probabilities for $\mu{=}0$ may seem surprising. If $\mu{=}0$, a model is deterministic. This can lead to unwanted cyclic behaviour, an artefact of the discreteness of the phase values, where very minor perturbations to phase are ignored due to rounding, leading to groups of oscillators staying unsynchronised forever. Similar phenomena have also been observed in other approaches used to model emergent synchronisation [10]. When some level of uncertainty is introduced to the model perpetually asynchronous cycles no longer occur.




(a) (b)

**Fig. 3.** Mean Phase synchronisation: synchronisation probabilities for different refractory periods, and synchronisation times for different rates of broadcast failure.

Figure 2b shows us that a higher refractory period results in shorter synchronisation times when the probability for broadcast failure is low. In general, a longer refractory period up to half the cycle length improves the rate of convergence to synchrony, which is consistent with the findings of [9]. Furthermore, for high values of $\mu$ the differences in synchronisation times for different refractory period lengths are negligible. Hence, a refractory period of slightly less than half the cycle, with a low coupling constant $\epsilon$, is optimal for this model of synchronisation. As $\epsilon$ is increased the results remain similar, but with a decrease in synchronisation times.

*Mean Phase Synchronisation Model.* We now instantiate the evolution and perturbation functions for a model of synchronisation similar to the work of Breza [5]. To that end, we set the evolution function to be the successor function, as in the previous section; however the perturbation function is more involved. In Breza's model, an oscillator perturbed by another firing oscillator updates its phase to be the average of its current phase and the phase of the firing oscillator (fixed as $T$ in our model). For this model of synchronisation there is no notion of coupling strength between oscillators, that is, $\epsilon$ is ignored. However our general oscillator model can still be instantiated to formalise such a protocol. We derive the following perturbation function: $\mathrm{pert}(\Phi, \alpha, \epsilon) = \left[\frac{1}{2^\alpha}(\Phi + T(2^\alpha - 1))\right] - \Phi$. Informally, the function calculates the result of iteratively taking the mean of the phase and $T$, for $\alpha$ iterations, and returns the difference between this and the original phase. Note that the perturbation induced by the firing of another oscillator is inversely proportional to $2^\Phi$.

We generate models for the parameter values examined for the M&S model of synchronisation, and again analyse the models with respect to $\varphi_1$ and $\varphi_2$. Figure 3a shows the synchronisation probability for different rates of broadcast failure and lengths of refractory period. It has similar characteristics to Fig. 2a.

That is, for almost all cases of $\mu$, the oscillators will always synchronise when the refractory period is less than half the cycle. Again as expected, $\mu = 1$ results in almost no synchronising runs, and $\mu = 0$ creates cyclic behaviour that leads to perpetual asynchrony. We can see that the Mean Phase (MP) synchronisation model is slightly more robust in this case, than a loosely coupled oscillator with the M&S synchronisation model. If we increase the coupling strength of the latter, however, it performs even better.

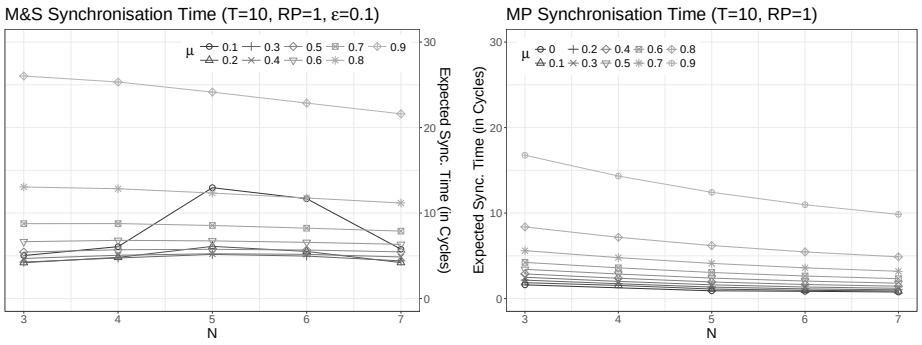
We now consider the time required to achieve synchronisation. Fig. 3b shows that, in most cases, a short (but non-zero) refractory period results in shorter synchronisation times. In general, it therefore seems optimal to choose a short, non-zero length refractory period. For low broadcast failure probabilities, however, there are negligible differences for refractory periods of different lengths. If we expect robust communication for a deployed network then we should choose a longer refractory period and so conserve energy.

*Network Synchronisation Scalability.* Figures 4a and 4b plot the synchronisation time against the population size for different rates of broadcast failure, for the M&S and Mean Phase synchronisation models respectively. For the M&S model we see that when $\mu > 0.3$, increasing the population size results in shorter synchronisation times, while a higher rate of broadcast failure yields longer synchronisation times. We conjecture that the surprising peaks for $\mu \leq 0.3$ are again an artefact of the rounding, resulting in cyclic behaviour, similar to that observed in deterministic models, as discussed for the M&S model. For the Mean Phase synchronisation model, we can again observe that a higher rate of broadcast failure yields longer synchronisation times. Similarly, increasing the population size results in shorter synchronisation times. However, in this case the rate at which synchronisation time decreases, given an increase in the size of the population, is more pronounced. Unlike the M&S model there are no peaks in the graphs indicating undesirable asynchronous cyclic behaviour. For the M&S model we observed that low coupling strength resulted in minor perturbations to phase being ignored due to rounding. In the Mean Phase model this does not occur, since the fractional part of the calculated mean phase is always $\geq 0.5$.

*Model Checking Scalability.* Using formal population models to analyse networks of indistinguishable oscillators is a promising approach. We checked networks with up to 7 oscillators, while to the best of our knowledge, other formal analyses using model checking turned out to be infeasible for more than four nodes for fully connected networks [13]. Memory and time used for model checking and construction of a single model, resp., are shown in Tab. 2. The increase in memory usage is as expected, and differences

**Table 2.** Memory for Model Checking (in MB) and Time for Model Construction (in seconds), for $T$=10, $RP$=1, $\epsilon$=0.1, $\mu$=0.1.

| | M&S | | MP | |
|---|---|---|---|---|
| $N$ | Mem. | Time | Mem. | Time |
| 3 | 124.63 | 0.09 | 131.30 | 0.09 |
| 4 | 161.33 | 0.37 | 162.42 | 0.43 |
| 5 | 262.62 | 1.65 | 261.39 | 1.61 |
| 6 | 592.94 | 5.28 | 610.20 | 5.42 |
| 7 | 1604.76 | 17.13 | 1495.59 | 16.88 |




(a) (b)

**Fig. 4.** Synchronisation times for different number of oscillators for Mirollo & Strogatz synchronisation (left), and Mean Phase synchronisation (right).

between the two models are relatively small. The properties can be checked in under a second. While our approach allows us to postpone the state space explosion problem, we cannot escape it completely. The major bottleneck is not the model checking time itself, but rather the model construction time. For individual models this was relatively short, but greatly accumulated for the parameter combinations we investigated, where thousands of models were constructed.

## 6 Conclusion

In this paper we presented a formal general model for networks of pulse-coupled oscillators, whose oscillation cycles are defined as a sequence of discrete states. We instantiated the general model for two different models of synchronisation used for the coordination of wireless sensor networks and swarm robotic systems. For each instantiation, and for a range of different values for model parameters, we automatically generated input for the probabilistic model checker Prism, encoded as a discrete time Markov chain. Finally, we used the results of model checking to analyse parametric influence on both the rate at which synchronisation occurs, and the time taken for it to occur; in particular, we discussed the trade-offs for parameter choices to minimise energy consumption in a network.

For future work, we intend to extend our current binary notion of synchronisation by introducing a metric, in the form of a reward structure, allowing us to reason about different degrees of synchronisation for global states. We also intend to formally encode energy consumption reward structures that will allow us to obtain quantitative results for those we reasoned about informally in this paper. A population model is appropriate when nodes are indistinguishable and the network is fully coupled. To analyse other network topologies, for instance a network of fully connected subcomponents, we could encode each such subcom-

ponent as a single population model, and take the cross product of the models for all subcomponents. To accomplish this it is likely that we would need to further refine our model, as this would greatly increase the state space.

## References


1. Alur, R., Dill, D.L.: A theory of timed automata. TCS 126(2), 183–235 (1994)
2. Amin, S., Elahi, A., Saghar, K., Mehmood, F.: Formal modelling and verification approach for improving probabilistic behaviour of robot swarms. In: Proc. IBCAST 2017. pp. 392–400. IEEE (2017)
3. Bartocci, E., Corradini, F., Merelli, E., Tesei, L.: Detecting synchronisation of biological oscillators by model checking. TCS 411(20), 1999–2018 (2010)
4. Bojic, I., Podobnik, V., Ljubi, I., Jezic, G., Kusek, M.: A self-optimizing mobile network: Auto-tuning the network with firefly-synchronized agents. Inf. Sci. 182(1), 77–92 (2012)
5. Breza, M.: Bio-Inspired Tools for a Distributed Wireless Sensor Network Operating System. Ph.D. thesis, Imperial College London (Mar 2013)
6. Buck, J.: Synchronous rhythmic flashing of fireflies. II. Q. Rev. Bio. 63(3), 265–289 (1988)
7. Bucur, D., Kwiatkowska, M.: On software verification for sensor nodes. J. Syst. Softw. 84(10), 1693–1707 (2011)
8. Christensen, A.L., Grady, R.O., Dorigo, M.: From fireflies to fault-tolerant swarms of robots. IEEE Trans. Evolut. Comput. 13(4), 754–766 (2009)
9. Degesys, J., Basu, P., Redi, J.: Synchronization of strongly pulse-coupled oscillators with refractory periods and random medium access. In: Proc. SAC 2008. pp. 1976–1980. ACM (2008)
10. Fatès, N.: Remarks on the cellular automaton global synchronisation problem. In: Proc. AUTOMATA 2015. LNCS, vol. 9099, pp. 113–126. Springer (2015)
11. Gainer, P., Dixon, C., Hustadt, U.: Probabilistic model checking of ant-based positionless swarming. In: Proc. TAROS 2016. LNCS, vol. 9716, pp. 127–138. Springer (2016)
12. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. FAC 6(5), 512–535 (1994)
13. Heidarian, F., Schmaltz, J., Vaandrager, F.: Analysis of a clock synchronization protocol for wireless sensor networks. TCS 413(1), 87–105 (2012)
14. Konur, S., Dixon, C., Fisher, M.: Analysing robot swarm behaviour via probabilistic model checking. Robot. Auton. Syst. 60(2), 199–213 (2012)
15. Kuramoto, Y.: Collective synchronization of pulse-coupled oscillators and excitable units. Physica D: Nonlinear Phenomena 50(1), 15–30 (1991)
16. Kwiatkowska, M., Norman, G., Parker, D.: Prism 4.0: Verification of probabilistic real-time systems. In: Proc. CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer (2011)
17. Mirollo, R.E., Strogatz, S.H.: Synchronization of pulse-coupled biological oscillators. SIAM J. App. Math. 50(6), 1645–1662 (1990)
18. Perez-Diaz, F., Trenkwalder, S., Zillmer, R., Gross, R.: Emergence and inhibition of synchronization in robot swarms. In: Proc. DARS 2016. Springer Tracts in Advanced Robotics, Springer (in press)
19. Werner-Allen, G., Tewari, G., Patel, A., Welsh, M., Nagpal, R.: Firefly-inspired sensor network synchronicity with realistic radio effects. In: Proc. SenSys 2005. pp. 142–153. ACM (2005)