Chapter 1

dummy-hack Introduction

Chapter 2

dummy-hack Background

Chapter 3

dummy-hack Tools and software

Chapter 4

Implementation

This chapter gives an overview of the developed musical multi-robot system. The main goal of the implemented system is to allow for individual musical agents in a musical multi-agent collective to interact with each other, in order to achieve emergent and co-ordinating behaviour—in our case synchronization—with varying degrees of self-awareness, collective-sizes, and of difficulty and certainty in the environment and communication. More specifically, the goal with the design is to enable the robot collective to achieve so-called *harmonic synchronization* within a relatively short time. Exactly what is meant by *harmonic synchronization* will be expounded in Section 4.4.

These goals firstly require of the agents the modelling of oscillators with their properties, like phase and frequency, as explained further in Subsection 4.1.1. To allow for interaction and communication between the agents, mechanisms so that the agents can transmit "fire"-signals, as well as listen for other agents's "fire"-signals, is necessary as well, and is presented in Subsection 4.1.2.

First, the system and the system components will be presented and introduced. Then, methods implemented for achieving the system target goal of *harmonic synchrony* in various synchronization objectives—firstly solely for oscillator-phases, then secondly for both oscillator-phases and oscillator-frequencies—will be described and presented.

Helpful way to read this chapter: how to achieve the final part when doing/using/performing the middle part using / to/with the first part?

4.1 The musical multi-robot collective

BESKR.: $\Big[$ about introducing, giving an overview, and describing the developed system (as well as the system sub-constituents — aka. what the system consists of). $\implies$ In a nutshell answering: What is the system, and what does it consist of? Why? $\Big]$.

Envision that we have a multi-agent collective scenario consisting of musical robots modelled as oscillators, solely communicating through brief "fire"-like audio-signals—greatly inspired by K. Nymoen et al.'s synchronizing "fireflies" [2]. These agents are initially not synchronized in their firing of audio-signals, but as time goes, they are entraining to synchronize to each other by adjusting

their phases and frequencies when or after hearing each other's audio-signals. An illustration of this is given in Figure 4.1.
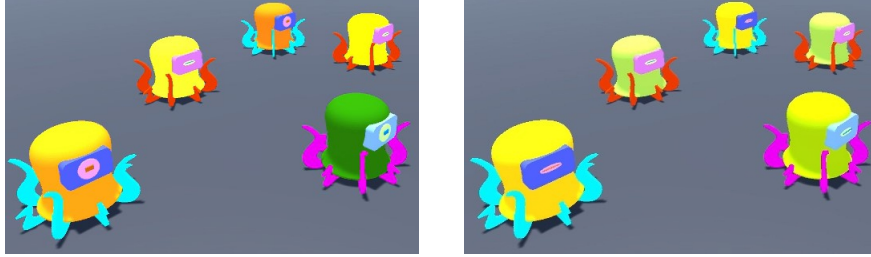
(a) The agents firing asynchronously at first. Here, only the two Dr. Squiggles with red tentacles are firing simultaneously, but the rest are not.

(b) Seconds later, after having listened to each other's fire-event signals and adapted themselves accordingly, the agents are here firing synchronously.

Figure 4.1: Decentralized synchronization of phases achieved in a musical robot collective, consisting of M. J. Krzyzaniak and RITMO's Dr. Squiggles.

These aforementioned audio-signals to be expounded further in Subsection 4.1.2, also referred to as "fire"-signals, are transmitted whenever an agent's oscillator *peaks* (i.e. after its cycle or period is finished, having phase $\phi(t) = 1$)—or actually every second *peak*, due to the target system goal of *harmonic synchrony*. All agents have the ability to listen for such transmitted "fire"-signals from their neighbours, which they then will use as a trigger to adjust themselves according to some well-designed update-functions to be elaborated upon in Section 4.2 and 4.3.

4.1.1 The individual agent: a musical robot

BESKR.: $\lceil$ about the individual agent with all its attributes and characteristics etc. (as e.g. an oscillator-component (cf. I.A, III.Intro., and 'Implementation' in Nymoen's firefly-paper)) $\rfloor$.

Notions like "agent", "robot", "firefly", and "oscillator" will be used interchangeably throughout the thesis.

4.1.2 Robot communication: the "fire"-signal

No individual agent is able to adjust or modify the state of any other agent, only its own, and the only means of communication between the agents will be through these "fire"-signals.

- Short and impulsive audio sound/signal, representing an agent's phase-climax and "fire"-/"flash"-event.

- Stigmergic co-ordination/communication.

- Facilitating PCO (pulse-coupled oscillators), different from phase-coupled oscillators with their differences.

- Aspekter fra Nymoens paper som presentert i Essay-oppsummeringen min.

4.2 Synchronizing oscillator-phases

BESKR.: | about various methods/algorithms/approaches (the nitty gritty) implemented to achieve the target/goal state of harmonic synchrony. $\implies$ In a nutshell answer: What different but comparable methods did you use to achieve harmonic synchrony with Phase-Adjustment? |.

If we first assume constant and equal oscillator-frequencies in our agents, we can take a look at how the agents adjust their—initially random—phases in order to synchronize to each other. This is then in contrast to the case in Section 4.3 where heterogenous and randomly initialized oscillator-frequencies in the musical agents are used and synchronized.

The goal state of the agents is for now to fire/flash simultaneously, after having started firing/flashing at random initially. Note that this is a different goal from the final and ultimate goal of *harmonic synchrony*. This is actually not true — as firing/flashing simultaneously technically would be considered having achieved harmonic synchrony. Hence, this "sub-goal" is actually a sufficient and weaker goal w.r.t. the main goal of harmonic synchrony.

In order for the musical agents to synchronize to each other, they will have to—due to their heterogenous and randomly initialized phases—adjust or update their own phases according to some well-designed update-/adjustment-functions, as presented below.

When it comes to the temporality and timing of when these updating functions are used and applied; Musical agents's phases get updated/adjusted immediately as "fire"-/"flash"-events from neighbouring robots are perceived.

4.2.1 Mirollo-Strogatz's "standard" phase-updates

One approach having been used to achieve this in the past is Mirollo-Strogatz's "Standard" Phase-adjustment in oscillators [1], as illustrated in Figure 4.2.

Each musical agent gets a new phase, $\phi(t^+) = P(\phi(t))$, accoring to the **phase update function** (4.1) upon perceiving a "fire"-event from one of the other musical nodes:

$$P(\phi(t)) = (1 + \alpha)\phi(t), \tag{4.1}$$

where "α *is the pulse coupling constant, denoting the strength between nodes*" [2], and t^+ denotes the time-step immediately after phase-climax. So, if $\alpha = 0.1$ e.g., then a musical agent's new and updated phase, immediately after hearing a "fire"-signal from another agent, will be equal to $\phi(t^+) = P(\phi) = (1+0.1)\phi = 1.1\phi$. 110% of its old phase ϕ, that is. Hence, and in this way, the agent would be "pushed" to fire sooner than it would otherwise (as nodes fire once they have reached phase-climax $\phi = 1$).

4.2.2 K. Nymoen's bi-directional phase-updates

This phase-adjustment, as in Figure 4.3, works very similarly to the phase-adjustment performed in the "standard" ***Mirollo-Strogatz*** approach presented earlier; the only difference being that now, nodes update their phases with the
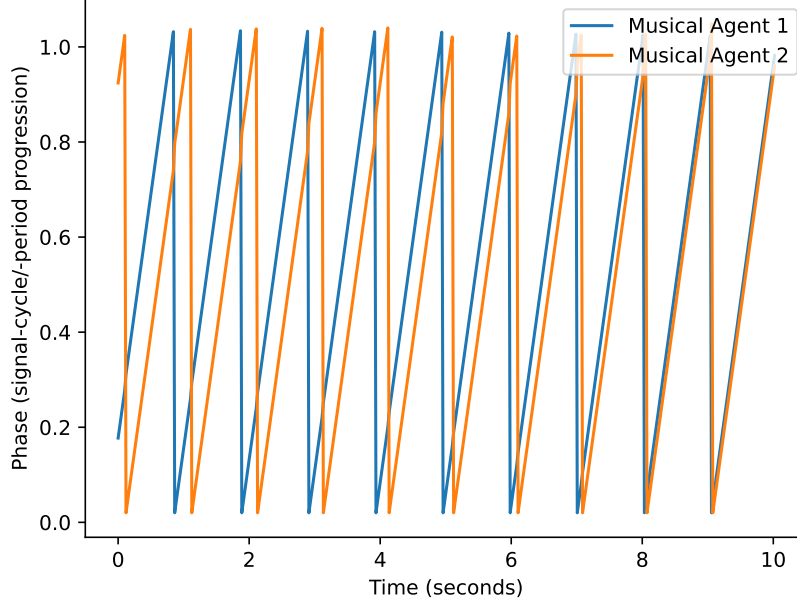
Figure 4.2: "Standard" phase-adjustment with Mirrollo-Strogatz's approach

slightly more complex **phase update function** (4.2) when hearing a "fire"-event from one of the other musical nodes — allowing for both larger, but also smaller, updated phases compared to the old phases:

$$P(\phi) = \phi - \alpha \cdot sin(2\pi\phi) \cdot |sin(2\pi\phi)| \tag{4.2}$$

The fact that new and updated phases can both be larger, but also smaller, compared to the old phases, is exactly what's meant by the Phase-adjustment being **bi-directional**, or as the authors call it in the paper as using "*both excitatory and inhibitory phase couplings between oscillators*" [2].

The effects then of adjusting phases—upon hearing "fire"-events, according to this newest update-function (4.2)—are that the nodes's phases now get decreased if $\phi(t)$ is lower than 0.5, increased if $\phi(t)$ is higher than 0.5, and neither—at least almost—if the phases are close to 0.5. This is due to the negative and positive sign of the sinewave-component in Equation (4.2), as well as the last attenuating factor in it of $|sin(2\pi\phi)| \approx |sin(2\pi\frac{1}{2})| = |sin(\pi)| = |0| = 0$, then if we have $\phi(t) \approx 0.5 = \frac{1}{2}$.

4.3 Synchronizing oscillator-frequencies

BESKR.: ⌈ about various methods/algorithms/approaches (the nitty gritty) implemented to achieve the target/goal state of harmonic synchrony. $\implies$ In a nutshell answer: What different but comparable methods did you use to achieve harmonic synchrony with Frequency-Adjustment? ⌋.
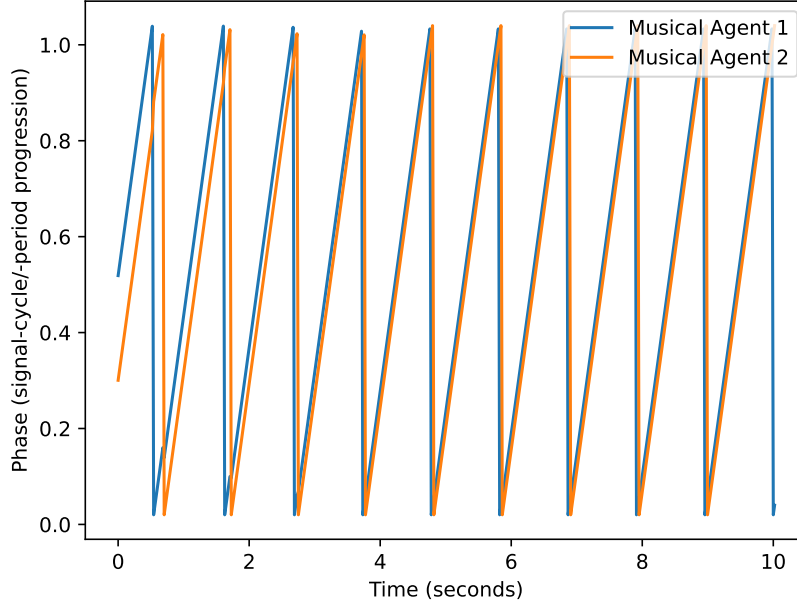
Figure 4.3: Bi-directional phase-adjustment with K. Nymoen et al.'s approach

Now over to the slightly more complex issue of adjusting and synchronizing frequencies—as well as phases—in the agent-/oscillator-collective. We introduce randomly initialized, non-constant, and heterogenous oscillator-frequencies in our musical agents — allowing in the music collective the playing of diverse rhythmic patterns as described further in Section 4.4. The agents are now then required to not only synchronize their phases to each other, but now also their initially different and random frequencies — so that frequencies are "legal" and *harmonically synchronized*.

The goal state now will not be exactly equal frequencies and phases, but rather the goal state of *harmonic synchrony*.

In order for the musical agents to synchronize to each other, they will have to adjust or update their own phases and frequencies according to some well-designed update-/adjustment-functions. Three approaches/methods that was implemented to achieve this, and the goal state of *harmonic synchrony*, is presented now. Notice the increasing degree of *Computational Self-Awareness* endowed/utilized/included in the methods.

4.3.1 Person X's low SA-leveled frequency-updates

A simpler strategy/method for achieving Harmonic Synchronization including Frequeny-Adjustment, specifically with no Self-Awareness capabilities, if it exists.

4.3.2 K. Nymoen's middle SA-leveled frequency-updates

This approach to Frequency Adjustment stands in contrast to previous approaches to synchronization in oscillators [fixed_freqs, fixed_range_freqs] where

the oscillators's frequencies are either equal and fixed, or where frequencies are bound to initialize and stay within a fixed interval/range.

In order to achieve this goal of *harmonic synchrony* in conjunction with—or rather through—frequency adjustment, we have to go through a few steps to build a sophisticated enough update-function able to help us achieve this.

When it comes to the temporality and timing of when these update functions are used and applied; Musical agents's phases get updated/adjusted immediately as "fire"-/"flash"-events are perceived, whereas agents's frequencies do not get updated until the end of their oscillator-cycle (i.e. when having a phase-climax $\phi(t) = 1$). This is also the reason why frequencies are updated discretely, not continuously. So-called H-values however, being "contributions" with which the frequencies are to be updated according to, are immediately calculated and accumulated when agents are perceiving a "fire"-/"flash"-event — and then finally used for frequency-adjustment/-updating at phase-climaxes.

Each agent i update their frequency, on their own phase-climax (i.e. when $\phi_i(t) = 1$), according to the frequency-update function $\omega_i(t^+)$:

$$\omega_i(t^+) = \omega_i(t) \cdot 2^{F(n)}, \tag{4.3}$$

where t^+ denotes the time-step immediately after phase-climax, $\omega_i(t)$ is the old frequency of the agent at time t, and $F(n) \in [-1, 1]$ is a quantity denoting how much and in which direction an agent should update its frequency after having received its nth "fire"-signal.

This is how we obtain the aforementioned $F(n)$-quantity:

4.3.2.1 Step 1: the "in/out-of synch" error-measurement/-score, $\epsilon(\phi(t))$

Describing the error measurements at the n-th "fire"-event, we introduce an Error Measurement function.

The Error Measurement function (4.4), plotted in Figure 4.4, is calculated immediately by each agent i, having phase $\phi_i(t)$, when a "fire"-event signal from another agent is detected by agent i at time t.

$$\epsilon(\phi_i(t)) = sin^2(\pi\phi_i(t)) \tag{4.4}$$

As we can see from this Error-Function, the error-score is close to 0 when the agent's phase $\phi_i(t)$ is itself close to 0 or 1 (i.e. the agent either just fired/flashed, or is about to fire/flash very soon). The error-score is the largest when an agent perceives a "fire"-signal while being half-way through its own phase (i.e. having phase $\phi(t) = 0.5$). We could also then ask ourselves, does not this go against the main/target goal of the system, being *harmonic synchrony* — if agents are allowed to be "half as fast" as each other?

"Hypothetical scenario": [

We could imagine a completely "legal" and harmonically synchronous scenario where two agents have half and double the frequency of each other. The agent with half the frequency of the faster agent would then have phase $\phi(t) = 0.5$ when it would hear the faster agent "fire"/"flash" — leading to its Error-score
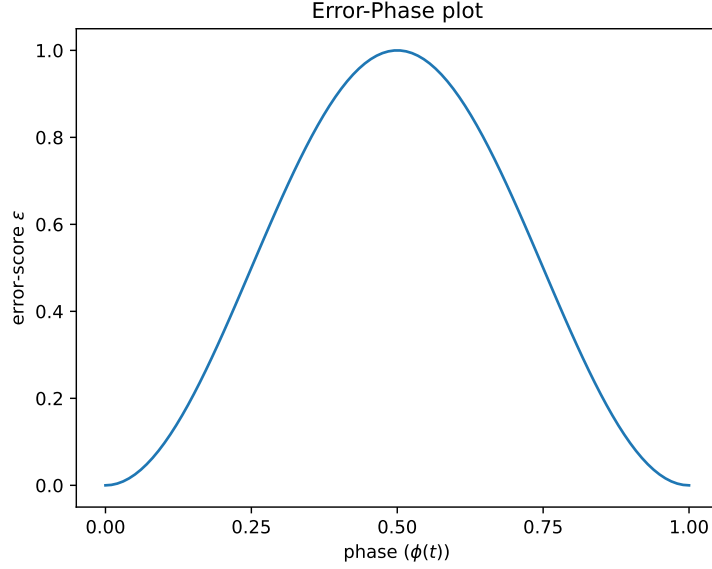
Figure 4.4: Error Measurement (4.4) plotted as a function of Phase

$\epsilon(0.5) = sin^2(\pi/2) = 1$, which then makes it seem like the slower agent is maximally out of synch, when it is actually perfectly and harmonically synchronized. This calls out for an attenuating mechanism in our frequency update function, in order to "cancel out" this contribution so that perfectly harmonically synchronized agents will not be adjusted further despite their high Error-measurement.].

This Error-Measurement/-Score forms the basis and fundament for the first component of Self-awareness, being the *self-assessed synchrony-score $s(n)$*.

4.3.2.2 Step 2: The first self-awareness component, s(n)

This aforementioned self-assessed synchrony-score, $s(n)$, is in fact simply the median of a list containing m Error-scores ϵ. Such a list is easily implemented in C# by declaring a List<float> called *errorBuffer* e.g. (i.e. *errorBuffer* is a list containing floating point values):

$$errorBuffer = \{\epsilon(n), \epsilon(n-1), ..., \epsilon(n-m)\}, \quad (4.5)$$

then leading to:

$$
\begin{aligned}
s(n) &= median(errorBuffer) \\
&= median(\{\epsilon(n), \epsilon(n-1), ..., \epsilon(n-m)\}) \in [0, 1],
\end{aligned}
\quad (4.6)
$$

10

where n is the latest observed "fire-event", and m is the number of the last observed "fire"-events we would like to take into account when calculating the self-assessed synch-score.

If we then have a high $s(n)$-score, it tells us that the median of the k last error-scores is high, or in other words that we have mainly high error-scores — indicating that this agent is out of synch. Conversely, if we have a low $s(n)$-score, indicating mainly low error-scores for the agent — then we have an indication that the agent is in synch, hence leading to low error scores, and in turn low $s(n)$-scores.

In other words, each agent hence has a way to assess themselves in how much in- or out-of-synch they believe they are compared to the rest of the agents. This is then the first degree/aspect of public[?] self-awareness in the design.

4.3.2.3 Step 3: frequency update amplitude- & sign-factor, $\rho(n)$

Describing the amplitude and sign of the frequency-modification of the n-th "fire-event" received. It is used to say something about in which direction, and in how much, the frequency should be adjusted.

$$\rho(\phi) = -sin(2\pi\phi(t)) \in [-1, 1] \tag{4.7}$$

For example, if an agent i has phase $\phi_i(t) = 1/4$, it gets a value $\rho(1/4) = -sin(\pi/2) = -1$; meaning, the agent's frequency should be decreased (with the highest amplitude actually) in order to "slow down" to wait for the other nodes. Conversely, if an agent j has phase $\phi_j(t) = 3/4$, it gets a value $\rho(3/4) = -sin(3/2\pi) = -(-1) = 1$; meaning, the agent's frequency should be increased (with the highest amplitude) in order to getting "pushed forward" to catch up with the other nodes.

Acts as an attenuating factor, when $\phi(t) \approx 0.5$, in the making of the H-value — supporting the goal of *harmonic synchrony*.

4.3.2.4 Step 4: the H-value, and the H(n)-list

The following value, being "frequency-update-contributions", is then (as previously mentioned) calculated immediately when the agent perceives another agent's "flashing"-signal:

$$H(n) = \rho(n) \cdot s(n) \tag{4.8}$$

Here we then multiply a factor $\rho(n)$ representing how much, as well as in which direction, the agent should adjust its frequency, together with a factor $s(n) \in [0, 1]$ of the adjusting agent's self-assessed synch-score. To recall, the self-assessed synch-score $s(n)$ tells an adjusting agent how in- or out-of-synch it was during the last m perceived "fire"-/"flash"-events — where $s(n) = 0$ signifies a mean of 0 in error-scores, and $s(n) = 1$ signifies a mean of 1 in error-scores. So then if this H-value is to be used to adjust the nodes's frequencies with, the frequency will then be adjusted in a certain direction and amount (specified by $\rho(n)$) — given that the agent is *enough* "out of synch"/"unsynchronized" (in the case $s(n)$ is considerably larger than 0).

The H-value says something about how much "out of phase" the agent was at the time the agent's nth "flashing"-signal was perceived (and then followingly

how much it should be adjusted, as well as in which direction after having been multiplied together with a sign-factor $\rho(n)$), given then that this H-value also consists of the *self-assessed synch score s(n)* — which again simply was the median of the list of error-scores, *errorBuffer* in our case.

We could look at this H-value as representing the direction and amplitude of the frequency adjustment weighted by the need to adjust (due to being out of synch) at the time of hearing "fire"-/"flash"-event n. Or in other words, this H-value is then the n-th contribution with which we want to adjust our frequency with.

Especially interesting cases are when we have $\phi(n) \approx 0.5 \implies \rho(n) \approx \pm 0$, as well as the last m Error-scores $\epsilon(n)$ being close to 0, also leading to $s(n) \approx 0$. In both of these two cases the entire frequency-adjustment contribution H would be cancelled out, due to harmonic synchronization (legally hearing a "fire"-/"flash"-event half-way through ones own phase) in the first case, and due to not being out of synch in the latter (having low Error-Measurements). Cancelling out the frequency adjustment contribution in these cases is then not something bad, but something wanted and something that makes sense. If these H-values then are cancelled out or very small, it is indicative of that nodes are already in *harmonic synchrony*, and hence should not be "adjusted away" from this goal state. On the other side, if these H-values then are different (e.g. closer to -1 and 1), it is indicative of that nodes are not yet in *harmonic synchrony*, and that they hence should be "adjusted closer" to the goal state.

All the calculated H-values are in my implementation accumulated and stored in an initially empty C#-list (of floats), referred to as $H(n)$, at once they are calculated. The $H(n)$-list is then consecutively "cleared out" or "flushed" when its H-values have been used for the current cycle/period's frequency adjustment (i.e. at the phase climax, when $\phi(t) = 1$), and is then ready to accumulate new H-values during the next cycle/period.

4.3.2.5   The final step: the frequency update function, $\omega_i(t^+)$

Now, we can pull it all together, for Nymoen et al.'s Frequency Adjustment approach for achieving harmonic synchrony with initially randomized and heterogenous frequencies.

When an agent i has a phase-climax ($\phi_i(t) = 1$), it will update/adjust its frequency to the new $\omega_i(t^+)$ accordingly:

$$\omega_i(t^+) = \omega_i(t) \cdot 2^{F(n)}, \tag{4.9}$$

where t^+ denotes the time-step immediately after phase-climax, and $F(n)$ is found by:

$$F(n) = \beta \sum_{x=0}^{k-1} \frac{H(n-k)}{k}, \tag{4.10}$$

where β is the frequency coupling constant, k is the number of heard/received "fire-event"s from the start of the last cycle/period to the end (i.e. the phase-climax, or *now*) — and the rest of the values are as described above.

This $F(n)$-value then, as we see in Equation (4.10), is a weighted average of all the agent's $H(n)$-values accumulated throughout the agent's last cycle.

4.3.3 Thorvaldsen's high SA-leveled frequency-updates

Min nye proposede algoritme for å oppnå Harmonic Synchronization med Frequency-Adjustment, som inneholder flere tilleggs- Self-Awareness-komponenter (e.g. Belief-awareness og/eller Expectation-awareness), sammenliknet med K. Nymoens tilnærming (jf. **Thorvaldsen's improved approach** og det jeg og Kyrre snakka om under 'Mid-November'-recall-notatet på 'reMarkable - Meetings - Kyrre' på reMarkablen). Jeg vil "beste" Kristian.

4.4 System target state: harmonic synchrony

As previously mentioned, this state of *harmonic synchrony* is then the system goal state which we want the musical robot-collective to end up in eventually, the sooner the better.

The state of harmonic synchrony is the state in which all agents in the musical collective "fire"/"flash" at an even underlying interval or pulse, a certain number of times in a row. This is not to say all agents will have to "fire"/"flash" simultaneously, as has traditionally been the case for pulse-coupled oscillators.

As one is creating an interactive music technology system, one might want to encourage and allow for the playing of musical instruments at slower and faster rhythms/paces, as it might be quite boring if all instruments were played at the exact same measure or pulse. As K. Nymoen et al. [2] reason when discussing their own interactive "Firefly" music-system, as well as coining the term of harmonic synchrony:

"*Temporal components in music tend to appear in an integer-ratio relation to each other (e.g., beats, measures, phrases, or quarter notes, 8ths, 16ths).*"

and

"*Being an interactive music system, people may want their device to synchronize with different subdivisions of a measure (e.g. some play quarter notes while others play 8ths).*"

Accomodating for these aspects, K. Nymoen et al. hence took inspiration from the concept of *harmonics* in the frequency spectrum of a waveform, in that each harmonic wave or signal has an integer-relationship frequency to the fundamental (smallest) frequency. This is exactly how the frequencies of the individual musical agents in our music-collective must be, in order for the agent-collective to be harmonically synchronized.

More accurately, all musical agents—in a harmonically synchronized state—will have frequencies $\in \omega_0 \cdot 2^{\mathbb{N}_0}$, where ω_0 is the agent with the lowest frequency's frequency (i.e. the fundamental frequency), and $\mathbb{N}_0$ is the mathematical set of natural numbers including the number zero. Hence, agents will typically have frequencies like $\omega_0 \cdot 2^0 = \omega_0$, $\omega_0 \cdot 2^1 = 2\omega_0$, $\omega_0 \cdot 2^2 = 4\omega_0$, or $\omega_0 \cdot 2^3 = 8\omega_0$. If all agents end up with these kind of frequencies, we say they have "legal" and harmonically synchronized frequencies.

4.5 Performance-measure: time until goal of harmonic synchrony is detected

BESKR.: "where I present the method used to evaluate the performance of the presented and newly proposed algorithms/methods (i.e. the Synchronization-/Performance-measure)" — Samuelsen's MSc-thesis I think .

Hvordan detektere harmonic synchrony.

Hvordan HSYNCHTIME er definert.

Node-firing-plot.