TIMENOT:

A COMPUTATIONAL NOTATION FOR TIME-ORIENTED LIVE CODING

By

ALEJANDRO FRANCO BRIONES, B.A.

Supervisor: Dr. David Ogborn

A Major Research Paper/Project

Submitted to the Department of Communication Studies and Multimedia

in Partial Fulfillment of the Requirements

for the Degree

Master of Arts

in Communication and New Media

McMaster University

Table of Contents

Este proyecto de investigación para obtener el grado de maestría se realizó con el apoyo del Fondo Nacional para la Cultura y las Artes, a través del Programa de Becas para Estudios en el Extranjero FONCA-CONACYT, 2018.

CULTURA FONCA

SECRETARÍA DE CULTURA

Abstract

TimeNot is a computational notation that can describe complex rhythmic ideas and poly-temporal structures, mainly for live coding practices. The notation has been implemented in two ways: as an extension to SuperCollider, and within the Estuary collaborative live coding platform. Concepts like duration, onset pattern, Euclidean rhythms, tempo canon and convergence point are used to described a time-oriented music that draws many of its core principles both from live coding practices and the works of Conlon Nancarrow.

1. Introduction

TimeNot is a computational notation that is capable of producing, in an expressive way, complex rhythmic ideas embedded in poly-temporal structures. The most relevant aspects of the notation draw its main features, particularly poly-temporal strategies for music creation, from the work of Conlon Nancarrow. The notation encourages performers to project musical ideas further from the present than in the conventional interaction model of live coding. By doing so, it critiques the infrastructure and time conceptions of live coding communities, seeking to widen the possibilities of their practices. The poly-temporal structures enabled by this notation form a space of resistance that might allow listeners and performers to experience time beyond the scope of accelerated and linear neoliberal subjectivity. The present project is an extensive attempt to produce a mode of performance that emerges from an experience and conception of time as slow, constant, in resistance, multiple, simultaneous, non-linear, digital/analogue, and rhythmic.

The notation is based on ideas previously implemented in the Nanc-In-A-Can SuperCollider extension, which was co-authored along with Diego Villaseñor de Cortina. In both that earlier project and the present work, the main strategy to produce poly-temporality is the tempo canon of Nancarrow. TimeNot allows the production of such canons but, in addition, it explores new and expressive ways of representing them by complementing the production of canons with strategies to describe other forms of temporal organisation such as global durations and an specific form of rhythm.

The notation has been implemented in two, different complementary ways with distinct advantages: It is embedded in Estuary (Ogborn, 2017) as a mini-language which allows it to engage in ensemble dynamics and be integrated in a rich ecosystem of languages that encourage diverse ways of thinking time and music. It is also an extension of the platform SuperCollider (Wilson, 2011) using its IDE and server taking full advantage of its sound-synthesis power and allowing it to be easily distributed.

2. Motivations and Goals

2.1 The problem of time

The way in which we conceptualise and experiment with certain temporal phenomena can define them as emancipating or alienating forces, capable of affecting the way in which we relate to the world. Therefore, it is necessary to reflect on and explore the ways in which we conceive time. Time is a key aspect in any imaginable social, political or cultural revolution. The twentieth century, as observed by Faramelli (2014), was determined by socio-political (and artistic) positions that might be described as vanguardist. Vanguardisms are highly essentialised, teleological and accelerated movements in which the ends justify the means. They tend to eventually reproduce the same patterns of oppression and alienation that motivated them initially. This pattern of teleology and acceleration has been integrated and weaponised in late capitalist societies (Deleuze and Guattari, 1972 & 1987). Hence, any meaningful socio-political and economic change needs to acquire a sense of time that transcends acceleration and linearity so that it simultaneously avoids the authoritarianism that often follows vanguardist revolutions, and avoids being captured by the dynamics of predatory market forces. An example is the environment of the creative industries and its exacerbation of the concept of “innovation” that feeds on the development of free software, the imagination of independent artists,

activists and other forms of creative endeavour. No matter how fast and “ahead” a community might get outside the scope of capitalism. It will either be captured and instrumentalised by market forces, or it will implode.

My research is motivated by the possibility of creating an artistic device that proposes an understanding of time that escapes accelerated and teleological forms of temporality – a computational notation that would favour a non-vanguardist conception of time in its use. The time that I am delineating is broadly influenced by: a) Concepts of nostalgia narrated by Boym (2001) and Wenzel (2006) which point towards a reflective and an anti-imperialist posture producing a relationship with the past beyond essentialism; b) Wajcman (2017) and Concheiro’s (2016) recognition of the accelerated rate in which our present lives are lived blurring the delimitations of work and free time as well as natural life cycles; c) Adam’s (2004) reflections on economic speculation and the financial market that impose to our current temporality a limited scope of possibilities and, coupled with linear historical, a particular certainty about the future, that inhibits our imagination and, ultimately, our capability of self determination; d) Rhythm as a cultural texture that point towards relationships between different agents in registers as the natural-social (Lefebvre, 2004), the digital (Davis, 2001) and the digital-artistic (Ikoniadou, 2014); e) It is particularly relevant Browne’s (2014) idea on a historiography based on poli-temporality and the tales of history posed in Zapatismo (Marcos, 2001, Valencia, 2010) in which a world that holds many worlds is possible that challenge formal teleology without dismissing historic processes.

The proposition in this project is a time that favours slowness over acceleration, privileges multiple and simultaneous timelines over one single homogeneous temporality, that might be understood as analogue as well as digital and that generally moves towards an uncertain trajectory that maintains an open sense of futurity.

My tools and practices as a researcher and artist are mostly embedded in an artistic context close to the international live coding community and in general to the field of live coding research, thus I designed this notation in order to intervene in this community.

2.2 Context

Live coding, in the present context, is the practice of writing programming code in front of an audience with artistic and/or pedagogic purposes (TOPLAP, 2010). Often the outcome of this code is a sonic or visual output that can be organised and understood as music or a live performance with dynamic visuals. One of the most relevant research questions in the live coding field has to do with the creation, appropriation and development of coding tools like DSLs, domain specific languages (Parr, 2007). Some DSL specialised in music creation are Chuck (Kapur et al., 2013), Tidal (McLean & Wiggins, 2010), FoxDot (Kirkbride, 2016), IxiLang (Magnusson, 2011), among many others. Each program capable of musical or visual production can determine its possibilities and strengths by developing not only its own specific functions (like rhythm and temporal structure production) but also the ways in which these functions are notated, described and expressed. Some languages might privilege a cyclic conception of time, like the platform Tidal (McLean, 2016), and some others could
exploit the power of recursion in relationship with time (Sorensen, 2010).

The work and ideas of Conlon Nancarrow provide another, equally important, context that helps properly frame this project. Nancarrow was a socialist, Mexican artist with a complex history as well as a unique exploration regarding time, media and music creation (Gann, 2006). Exiled from his birth-country given his involvement in the Spanish Civil War on the side of the socialist government and marginalised from the overwhelmingly nationalist Mexican music elite academic scene for sounding “too American” he decided to explore the rhythmic and temporal possibilities of the Player Piano,

opening a whole new territory of musical creation and produce concepts that allow poly-temporality to be organised in imaginative and creative ways.

Nancarrow never intended to make his work public or accessible; the approximately 50 piano player studies he produced are motivated by his intellectual curiosity and his vast imagination. Nevertheless there is a rising interest in his music by academia, global classic music elite, engineers, intellectuals, experimental artists and music journalists like György Ligeti, Charles Amirkhanian, Gerhard Trimpin, Rodolfo Halffter, among many others. Such interest led to the acquisition of every relevant document related to the work of Nancarrow by the Sacher Foundation in Switzerland. This situation has dramatically restricted and inhibited any research or knowledge production based on the direct work of Nancarrow. Nowadays it is only accessible to people who can afford mobility to its location in Europe, and that can claim legitimacy on heavily institutionalised settings. This presents a particular challenge to people from the global South, particularly Mexicans.

Given the private nature of the medium Nancarrow favoured to compose his works, the player piano, it is very difficult to reproduce it in conditions that resemble the ideal listening situation of this music. Dominic Murcott (2014) has pose the question of reproducibility to reanimate the music of Nancarrow. Regardless of precise documentation as the Ampico Records and other audio and video recordings, there is an identified need to present Nancarrow's music in live conditions in order to keep it as part of the conversation on music experimentation. The notation and the software that I have developed aims to distance itself from the documents, scores and direct production of Nancarrow (as captured by the Sacher Foundation) and rather seek to remain as faithful as possible to the ideas motivated and produced by the artist. Simultaneously, this notation would animate Nancarrow's work for other communities, in this case the live coding community, and propose a well suited media to allow poly-temporal canonical music strategies as the ones of Nancarrow.

The poly-temporality that Nancarrow proposed can be regarded as highly algorithmic; in it, temporal and pitch mapping functions are the basic principle. This means that the melodic material of any given music work (which consists mainly of pitch and duration series) can by transposed into any tempo or any pitch register. Nancarrow developed the concept of a convergence point (CP) as a way of organizing this highly fungible musical material. The CP is a point in time in which the formal and the chronological temporalities of a musical idea are identical. According to Thomas (1999), this strategy allows us to listen different timelines moving towards the same point in chronological and formal time. Given that a poly-temporal canon can be generated very easily by algorithms, it is a musical strategy that can be implemented in computational settings with relative ease. Collins (2003, pp 1) describes a compositional system capable of producing various kinds of tempo canons, a precedent for this notation. More recently, I have developed, along with the programmer, musician and philosopher Diego Villaseñor, the software Nanc-in-a-Can Canon Generator (Franco & Villaseñor, 2018). Nanc-in-a-Can Canon Generator provides a programming notation that allows musicians, artists, programmers and other creative users of code to create multiple and simultaneous sound timelines based on the work of Conlon Nancarrow. This software was developed in SuperCollider (McCartney, 2002) because of its powerful computational and synthesis capabilities and its extended use among live coding practitioners in Mexico City. The following example of Nanc-in-a-Can usage allows the user to create a major scale in three different tempos that converge at the fifth event (G, midinote 67).

```
(
// convergence canon;
~conv= Can.converge(
        melody: Can.melody(
        [8,8,8,8,8,8,8,8].reciprocal, // 1/8 rhythmic figures
        [60,62,64,65,67,69,71,72]),
        cp: 5,
        voices: Can.convoices([50,72.5,75],[-12,0,12])
        ).visualize(s)
);
```

Code Example 1: The lines of code above show the possibility of synthesising several and multiple computations designed to facilitate the production of temporal, convergence canons.

What I find more interesting are the “human-to-human” communication aspects of this notation. This can be referred to as the cognitive dimensions of computational notation (Green, et al. 2001), an idea fundamental to produce software interfaces that emphasise (human) cognition as much as computation. One cognitive dimension of notation according to Green is role-expressivenes, which denotes that “the purpose of a component is readily inferred (Green et al., 2001, pp. 345)”. The notation shown in Code Example 1 suggests the words Canon of Convergence in the text Can.converge given that this code creates a tempo canon that converges. Thus, making this notation highly role-expressive. Moreover, this piece of code not only executes a series of computations that the program performs but it also brings to the foreground a series of cultural references that locate the user of the program within a cultural and socio-political framework. This framework is the work of Conlon Nancarrow and its most salient temporal strategy to establish poly-temporal musical structures: Tempo Canon. The word ‘can’ also makes reference to the rigid steel or tinplate container normally used to preserve food which often is related with vulgar or industrialised products and, in this case, cheap nourishment for survival. This notation attempts to re-contextualise and re-appropriate musical strategies often related with a privileged music elite, often related to academic activity, that claimed the ideas of Nancarrow as part of their tradition even though the Mexican socialist artist required an untraditional context to flourish.

The canon generator, by distributing it in communities away from the people that often control the canon of the music academy, allows new meanings for the work of Nancarrow.

The notation style of Nanc-in-a-Can is heavily embedded in SuperCollider's inherent notation and it has a limited scope. There are some aspects that do not facilitate live coding performance, for instance: the nested parenthesis very particular to SuperCollider. SuperCollider language is not an expressive notation for rhythm as time, in this context, is mostly represented through inter-offset durations, namely wait patterns as understood in SuperCollider's Pattern Library (Harkins, 2009). This action produces a wait pattern, which is expressed as duration, obfuscating its meaning and compromising role-expressiveness. A more transparent use of the duration could refer to the total length of a musical idea (which cannot be expressed easily in the SuperCollider Pattern Library) or the length of a sound event (which is expressed as legato). Rhythm is more intuitively described in terms of offsets and onsets over an underlying grid, which can hardly be inferred by the duration value described in SuperCollider Pattern library. Likewise, it requires a binary distribution to be installed, which is an impediment in more pedagogic settings, differently from Estuary which requires zero installation. From this starting point I have identified the need for a new notational system that might be capable of representing the novel and compelling temporal forms already made possible by Nanc-in-a-Can.

3. Fundamental Aspects of the Notation

3.1 General Remarks

```
|. 4s .| xxxxxxxx
ra: 4:5:6 tr: 0|12|24 cp: last
synths: saw sqr tri pitch: 60 62 64 65 67 69 71 72
```

Code Example 2: Full use of TimeNot

The example shown in Code 2 shown above reproduces the scale of Example Code 1 and presents a program that exemplifies all the main possibilities that TimeNot allows: a sequence of global durations and a rhythm arrangement in line one, a canonic configuration in line two, and a configuration of instruments and pitch in line three. The interplay among these four components produces rich poly-temporal sonic textures. When this example is executed it produces a series of musical events extending from the moment of evaluation into the future (a C major scale repeated in three octaves with the temporal proportions of 4:5:6 over a total duration of 4 seconds).

The creation of this notation aimed to integrate rhythmic strategies with the aforementioned canonic ideas in such a way that the nuance and specificity of temporal organisation produce a fertile ground for time-oriented sonic experimentation. The main idea approached in this notation is the tempo canon. However, many additional rhythmic strategies and techniques can be notated easily.

A fundamental decision was whether to produce it it as an extension of Nanc-In-A-Can Canon Generator or, as was ultimately decided, to produce an independent software package based on the algorithms found in the Canon Generator. If I were to have produce an extension of Nanc-in-a-Can I would have been able to concentrate further on the notation and wouldn’t had to re-implement the algorithms necessary to produce the canonic data. However, proceeding in that way, the software would have been completely dependent on the SuperCollider platform.

I opted instead to create a new, independent, software project based on the ideas of Nanc-In-A-Can so the notation could be adapted to many contexts. TimeNot is now available in two forms: within the Estuary platform, and as a SuperCollider extension. Estuary (Ogborn et al., 2017) is an experimental software that can be defined as a platform for learning and creating using live coding as a main strategy that favours a multilingual approach. SuperCollider (McCartney, 2002) has a powerful audio synthesis engine as well as a well established community of users around the world. Both platforms respond in different ways to my cultural and social context; these are the best ways to give access to broader and more general users already introduced to live coding and also key developers and specialised audience.

There are other positive aspects of this decision; the separation of TimeNot and Nanc-In-A-Can Canon Generator allowed Diego Villaseñor to develop a notational style embedded in SuperCollider called Fluent Can, a parallel and independent project of this one that responds to the idiosyncrasies of SuperCollider, thus extending the software and proposing new ground of exploration that respond to an original premise of Nanc-in-a-Can: to design a software that can be self-contained in SuperCollider. Therefore allowing its integration to a broad ecosystem of software development and practices already well established. In other words, a software that can be easily adapted to the live coding ecology of software experimentation and production in a global scope, however with particular emphasis on Mexico City.

Interestingly, the parallel development of Fluent Can and TimeNot favoured a dialogue with tensions and resistances between Villaseñor and myself that ultimately proved to be very fertile for new paths of exploration. For example, ideas like the rhythmic notation (see 3.2.2) proposed in TimeNot were implemented in Fluent Can and ideas like the period of Fluent Can were implemented in TimeNot

(as duration, see 3.2.1). The next four subsections each review a specific “sub-notation” of TimeNot in more detail.

3.2 Strata.

The architecture of the notation is organised into four main strata or sub-notations. Three of these allow the user to organise sound events in time . The last stratum aids the organisation of sound parameters.

The three time-oriented sub-notations reflect three main temporal strategies that imply a heterogeneous understanding of temporal relationships. The first one allows the user to embed the sound output in a specific overall duration; this allows the user a very intuitive degree of control over the music material. The second sub-notation provides an expressive and comprehensive way to create rhythmic structures; this aspect of the notation was identified as the one that diverges the most from the possibilities endemic to SuperCollider’s notational style. The third sub-notation is the stratum in which rhythmic ideas are transformed into a canon. This is the part of the notation in which the tempo canon ideas developed by Nancarrow are put into practice. The last sub-notation provides a simple syntax to invoke instruments and organise its parameters into different kind of patterns.

3.2.1 Duration Notation

Having control over the global duration of the canonic/rhythmic structure helps to have an overal view of the result. This stratum represents time as a succession of events that do not favour detailed categorisation or differentiation among its internal components. Time here corresponds to the concept of durèe (Bergson, 2002) allowing users to produce simple sequences of events producing an ever-going sense of becoming.

The duration of the rhythmic/canonic structure is determined by a number followed by an *s* that represents seconds. However, the number could represent beats in a given tempo by adding a *t* or

indicate amount of cycles (cps) by adding a *c*. These multiple ways of expressing duration respond to the multi-contextual nature of this notation.

To determine a duration the number should be embedded in a special list that uses special symbols and separators. The lists that uses the |*: :*| as a delimiter and the | as an internal separator generate a musical idea. It suggests a loop that will be implemented in a future version of the notation. An unlooped event is delimited by |. .|. A finite number of repetitions can be expressed with the symbol *%* followed by whole number that determines the number of repetitions. If the duration of the structure is omitted, the default is a 2 second event.

The first line of example code 3 produces a four second long event, the second changes the total duration of the canon from 4 to 8 seconds, the next one produces a 1 second structure that is repeated 8 times and the last one produces a sequence of three structures with the corresponding durations in a set of events.

```
|. 4s .| x
|. 4sM2 .| x
|. 1s %8 .| x
|. 2s | 1s | 3s .| x
```

Code Example 3: Duration

As can be observed in the second line of example code 3, it is easy to change the metric depth of a rhythmic structure without losing reference to the original duration, the notation required is M2, M4, M8 to multiply the duration by 2, 4 and 8, and m2, m4, m8 multiplies the duration by 0.5, 0.25, 0.125.

3.2.2 Rhythmic Notation

The rhythm strategies and techniques that inspired this sub-notation respond mainly to four different theoretical positions regarding rhythm. a) The notation responds to the definition of music provided by

Arom: “a succession of sounds capable of giving rise to a segmentation of time during which it flows in isochronous units” (1991, pp. 11). This definition suggests a profound relationship between time and music that holds true throughout his analysis of the music of Central Africa. b) The conversations regarding time and rhythm I had with the Mexican composer Germán Romero over many years have been greatly influential for the development of various ideas regarding time. c) The implementation of Bjorklund algorithms to analyse and frame popular music proposed by Toussaint (2003). d) Finally, the music of Damaso Pérez Prado, a Cuban-Mexican big band director and composer who has developed a broad vocabulary on Afro-Caribbean music and Euclidean rhythms deeply rooted in Mexico City. Pérez Prado represents an integral part of my own cultural background. In the rhythmic style of Pérez Prado the social, cultural and aesthetic notions of rhythm found in African and post-diasporic (Gilroy, 1993) music traditions persist. An isochronous segmentation of time is produced with the onset patterns (see 3.2.2.1) that responds to (a) and (b), a sub-notation for repetition allows periodic ideas (see 3.2.2.2) and sub-notation for Euclidean patterns was provided as a way to create music based on (c) and (d).

The rhythmic aspects of the notation can be used independently from the canonic ones. Nevertheless a minimal rhythmic idea has to be written for the notation to produce sound. An example of a comprehensive use of this rhythmic notation might be the idea in code example 4. In this example an opening idea is presented in 5/8 manually introducing rhythmic onsets and offsets (see 3.2.2.1), then it is concatenated to an Euclidean pattern representing a *tresillo* over a two attack onset pattern (see 3.2.2.3) that is repeated two times (see 3.2.2.2), finally another 5/8 idea is manually expressed to close the musical idea.

```
|. 8s .|
xoxxo !3:e:8 p: xx #2 xooox  samples: hibongo
```

Code Example 4: Example of a full Rhythm Pattern

3.2.2.1 Onset Patterns

Notating minimal rhythmic ideas as onset patterns has two main advantages; it provides a sense of materiality to the music that respond to an almost embodied representation of an onset. It also frees the user from being limited to algorithmic structures (such as Euclidean rhythms), allowing arbitrary organisations that do not respond to explicit logical or computational patterns.

Patterns are notated with the characters *x* and *o*, where *x* denotes an attack and *o* denotes a rest. In the following example, the first pattern produces two attacks followed by 6 rests, the second pattern produces the Cuban *clave* rhythm and the last produces a ternary pattern with all attacks in the first three eighths, but only the first two in the second group of three eighths. This last pattern is the main rhythmic figuration of the ostinato in Pérez Prado's Voodoo Suite.

```
xxooooooo xxooooooo samples: hibongo;

xooxooxo ooxoxooo samples: lowbongo;

xxxxxo            samples: bd
```

Code Example 5: Example of Onset Pattern

3.2.2.2 Repeat Patterns

Repetition allows the user to produce meta-metric cycles in which the same idea is presented until a variation marks the end of a period. This can be achieved because the onset patterns and the repeat patterns can be composed together to express a single musical idea.

With the symbols *!* and # we can indicate how to repeat a pattern. It is possible to create nested ideas within this language, this means that a repeat pattern can contain an onset, repeat or eEucidean pattern. After the *!* the onset, repeat or euclidean pattern to be repeated should be written and after the # a value that represents the number of repetitions. The first line in code example 5 can be simplified as shown in the first line of code 6. In the second line of code 6 the "Voodoo Suite ostinato" is repeated three times and a meta-metric idea of four repetitions is marked by a variation on the main idea.

```
!xx!o#6#2  samples: hibongo

|. 4s .|
!xxxxxo#3xxoxxo synths: tri pitch: 60 63 65 67 60
```

Code Example 6: Example of Repetition Pattern

3.2.2.3 Euclidean Patterns

The Euclidean Pattern sub-notation is a very expressive and complete tool for rhythm which can integrate embedded onset, repetition and other euclidean patterns. The non-optional values to be given are n and k values explained in the paper by Godfried Toussaint (2003). The Euclidean Algorithm produces patterns of distribution of integer numbers as even as possible. Given an n number of time intervals and a k number of impulses, this algorithm provides a simple way to distribute the impulses over the time intervals. These patterns are found in many forms of music, particularly “in sub-Saharan African music, and world music in general” (Toussaint, 2003, pp. 1). The syntax proposed in this notation is *k:e:n*, a number representing impulses and another representing intervals. The first line of code example 7 generates a Cuban *tresillo* and the second generates a *clave* that is the combination of a *tresillo* concatenated to a 4/4 measure with onsets only in the second and third beats.

```
3:e:8 samples: hibongo
3:e:8 ooxoxooo samples: hibongo
5:e:8:r:4 p: xx  samples: bd
```

Code 7: Example of Euclidean Patterns

The k and n values are chained by the operator *:e:;*There are three other ways to manipulate the structure produced by this sub-notation: *:r:* which creates a rotation value, this means that the pattern produced will maintain its same number of impulses and same intervals but this would appear in a different point of the structure. The operator *p:* produces a structure that can be a repeat, an onset or another euclidean. This pattern becomes the k value and the non-k values of the time intervals, a series of offsets occupying the same length as the pattern are produced, if k is *xx* the non-*k* will be *oo*, as in the example above

3.2.3 Canonic Notation

A tempo canon refers to an Ars Nova technique whereby a literal repetition of a melodic idea is produced but with a tempo (and pitch) transposition. In other words, a melodic idea is repeated simultaneously in another register and with another tempo (Gann, 2006). A fundamental component of the tempo canon is the convergence point (see 3.2.3.2). Such tempo canons can be organised according to Cowell's notion of harmonic rhythm (1917 [1996]). Harmonic rhythm, in this sense, is organising rhythms in proportions that recall the harmonic series. For instance, a rhythmic relationship of 2:3 is analogous to an interval of a fifth in pitch organisation.

A fundamental difference between the tempo canons produced by the earlier Nanc-In-A-Can project and those produced by TimeNot concerns the minimal components of a tempo canon. Conlon Nancarrow was restrained by the media at hand: the player piano. The player piano does not allow to play two or more notes that are the same, in the same register, at the same time. To perceive distinct tempos it was necessary to differentiate the voices of the canon somehow. This was achieved by the pitch transposition. In TimeNot, where an alternative mechanism to differentiate voices is provided, pitch transposition is not as fundamental an aspect of the tempo canons. In TimeNot pitch transposition value is not a mandatory parameter for canonic transformation. Differentiation of voices is acquired by providing a straightforward mechanism to change instruments per voice. Different voices can be identified by different timbre qualities instead of by pitch registers. Canons can be thought of as based on samples (as is also currently favoured in Estuary). In TimeNot, pitch transposition is optional in the case of pitched instruments and, in the case of unpitched (sample-based) instruments, it is incompatible. In this way, less typing is needed to produce a "canonic transformation" and the idea of a tempo canon becomes even more focused on temporal aspects of the musical ideas.

The model that is implemented here so far is what Nancarrow would have called a “convergence canon”, that is to say a canon with only one CP and without tempo change per voice.

In TimeNot, the number of voices per canon is decided by the number of values given to the argument ratio. Ratio is a list of proportions which can have a corresponding list of transpositions that ‘canonise’ the rhythmic structure. The arguments for the canonic transformation are explained in 3.2.3.1 and 3.2.3.2.

3.2.3.1 Ratio and Transposition

Ratio key is a list of tempos separated by *:* which indicates the proportional nature of the tempos produced. A set of ratios as *3:4* would produce a relationship of three sounds in one voice every 4 in the other, similarly to the well known poly-rhythm common in various forms of music in which 4 eight notes play at the same time than a triplet of eight notes. Transposition is a list with | as a separator that produces a transposition value in midinotes. This argument produces an interval of pitch that will be assigned to each voices. Transposition is a pitch interval that depends on a pitch value (see 3.2.4.2) which default is a sequence of notes (a melody). If no value is given the default is midinote 60 (central C of the piano), hence the transposition interval will pivot from this point. Transposition has a default of 0 since unpitched instruments will not need it.

3.2.3.2 Convergence Point

The convergence point is the instant in which the chronometric time and the structural time of each voice in a convergence canon is identical. This means that the event position of the canonised rhythmic ideas converges. Some keywords are accepted, like *last* that will provide a CP in the last attack, *palindrome* will provide a CP in the event closest to the middle of the structure.

```
|. 35s .| xoox!5:e:8#5xxoxoooxx
ra: 13:17:20 tr: 0|5|8 cp: last
synths: saw sqr tri
eu pitch: 60 67 65 68 72 75.5 67 55 56 59 60

|. 12s .| !xxoxox#4 ra: 1:0.75:1.65 cp: last samples: cabasa hibongo oh
```

Code Example 8: Example of Two Canons

3.2.4 Sound Notation

3.2.4.1 Instruments

The instrument key is a list of instrument names. Instruments are organised as lists that will cycle through each voice of the canon. This means that the first instrument will be used in the canonic voice 1, second instrument in canonic voice 2, etc. If the number of voices is greater than the number of instruments, instruments for the additional voices are assigned by wrapping around to the beginning of the list again, as necessary. For the moment there is no notation to express the idea of assigning multiple instruments to a single voice. It is necessary to indicate the type of instrument with the keys: *synths:* and *samples:* in the case of the SuperCollider extension and *dirts:* in the case of the Estuary implementation.

3.2.4.2 Other sound parameters

For the moment there are two additional parameters: pitch and amplitude. There are two main ways to organise these parameters: the key *iso* will generate an isorhythmic relationship between the rhythmic values and parameters. In other words the provided parameters will cycle as a sequence from left to right as many times as is necessary to cover all the rhythmic values. The key *eu* produces a Euclidean distribution of parameter values, taking into consideration the rests and sounds of the rhythmic structure.

```
xxxxxxxx eu pitch: 60 64 67 72

xxxxxxxx iso pitch: 60 64 67 72
```

Code Example 9: Parameter organisation. Difference between eu and iso

Pitch and Amplitude

There are two kinds of instruments: Pitched and unpitched. The sample-based instruments normally do not have pitch, so declaring pitch will do nothing when using one of these. The unpitched instruments are sample-based and a similar key is provided to this instruments: *rate.*

The key *amp* will produce a pattern of amplitude values normalised from 0 to 1 using SuperCollider's model of amplitude treatment. The amp values will be the same for every voice of the canon.

```
|. 4s .|
!x#30 5:e:8 p: xo samples: hiconga eu amp: 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8
```

Code Example 10: Amplitude with parameter control eu

4. Implementation

The present project is distributed as free software in github repository with a Copy Left License. The notation has two implementations: as a SuperCollider extension and as a live coding language within the Estuary platform. The free and open source software licensing of this project allows musicians and programmers to access and modify it according to new or personalised necessities and experiments. The source code and every item produced as part of this notation is free and open source software, released under the terms of the GNU Public License (version 3).

4.1 Haskell Core

The core of the program was written in the programming language Haskell (Lipovaca, 2011); this is a purely functional language that has strong support for combinatorial parsing, a style of parsing that enables the programmer to compose broad parsers by combining many smaller ones modularly. Strong

typing allow TimeNot to be easily reasoned, proved and deduced which provides a framework in which conceptualisation becomes a meaningful and integral part of the programming process. Similarly, its inherent laziness (a program will only be executed when it is forced to show a result) allows programmers to think of virtual structures in permanent state of transformation, a quality that invites for further formal exploration. The tension between an untamed and baroque programming style as my own and Haskell's inherent concise elegance is also a source of tensions and resistances that are though-provoking and interesting. Haskell has been used in other projects related to live coded music like McLean's (2010) Tidal, or Bell's (2011) Conductive which pose interesting questions over time related to this project. Estuary's main language for development is Haskell, which facilitates TimeNot to be incorporated as a dependency (see 4.3). Helper functions were produced that allow communication between the Haskell core and SuperCollider, and the incorporation of TimeNot into Estuary as a dependency. Both implementations depend on the Haskell core here described.

4.2 SuperCollider Implementation

In the SuperCollider implementation of TimeNot, the SuperCollider integrated development environment (IDE) is used as an editor that communicates via Open Sound Control (OSC) (Wessel & Wright, 2002) with the core program written in Haskell. At the same time, an extension is provided that allows to use SuperCollider's server to support the creation and allocation of digital instruments and as a synthesis engine. The class TimeNot provided in the repository of the project allows the IDE to identify a key word provided by the live coder, that, when found, causes the text in the environment to be sent to a Haskell REPL (read-evaluate-print-loop) process to be parsed and processed. This allows to have editor windows dedicated to TimeNot while others are used as conventional SuperCollider files

making it possible to combine Patterns, ProxySpace and other modalities of sound productions native to SuperCollider.

The major advantage of the SuperCollider implementation of TimeNot is SuperCollider's highly performant synthesis engine. Once the canon structure produced in Haskell is transformed into individual sound events, the parameters of those events are sent back to SuperCollider to be transformed. SuperCollider has a robust infrastructure to produce sounds that includes virtually every possible sound synthesis process. Moreover, SuperCollider is a widely used platform that thus provides another entry point for a broad number of potential users of the new notation. A challenge for the development of TimeNot is the discrepancy between the time conception used in TimeNot (UTCTime) and the System clock native to SuperCollider which makes potentially unstable the synchronisation between TimeNot and SuperCollider and an unclear framework to allow network or ensemble performances. A clear disadvantage of SuperCollider are the multiple complication to install the software in itself and its adequate extensions for people unfamiliarised with programming practices and the complicated configuration system between hardware and software.

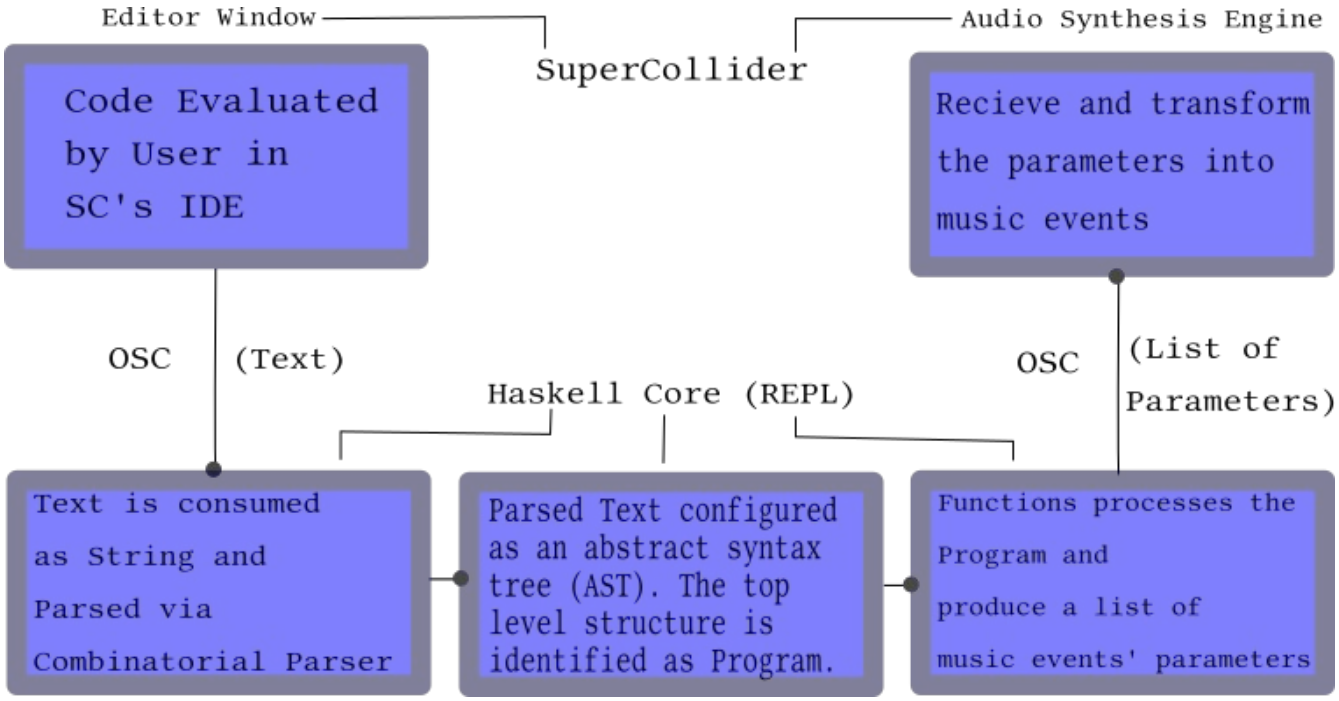

Diagram 1: TimeNot as a process in the SuperCollider's implementation

4.3 Estuary Implementation

The second implementation of TimeNot is as a mini-language within the larger Estuary collaborative live coding platform (Ogborn et al., 2017). Estuary is a multi-lingual and multi-contextual live coding software allocated online. There are three characteristics of Estuary that interest me concerning the implementation of this notation. Firstly, Estuary is a web-based program thus it requires zero installation and it is potentially accessible for anyone with internet access. Secondly, the languages implemented in Estuary can be used in ensemble conditions. This is relevant because creating canonic forms is intuitively understood as a compositional task, when it is possible to think of ensemble it acquires new connotations that is worth exploring. Thirdly and most important, it is multi-contextual, which means that many live coding languages can be used simultaneously. Thus, either one performer could potentially use multiple languages in the same performance or multiple performers could use multiple languages. In both cases, novel sonic possibilities emerge and new conceptual or musical assemblages can be imagined.

The multi-contextual nature of Estuary allows TimeNot to interact with other live coding mini-languages contributing to the rich ecology of different specific purpose languages that inhabit the software. Estuary uses WebDirt, a engine for sample playback modelled on McLean's Dirt created as part of Estuary. With WebDIrt it is possible to invoke the sample library that Estuary uses. For the moment, TimeNot is not tacking full advantage of this tool, but WebDirt allows the notation to be extended in this direction: implementing effects and processes from WebDirt. A disadvantage of implementing TimeNot in Estuary is the computation performance is less robust than SuperCollider. Thus, it is more restrictive in complex sound synthesis production, it is difficult to extend via OSC and also lacks a multi-channel output system.

5. Future Work and Conclusion

The software here described presents various clear areas of development. These further developments would reflect more clearly the initial premises of this project and would, as well, as pose new interesting territories for exploration for this project. These are some of the areas identified:

- The rhythmic notation needs to produce recursive structures. In other words, the possibility to have nested rhythms as in McLean's Tidal Cycles (2016) or in the rhythmic ideas of Brian Ferneyhough (Paetzold, 2010). The rhythmic stratum of the program will become central if it can generate recursive durations and represent aspects of sound other than onsets and offsets. For instance, writing information that could only affect a specific voice instead of the whole canon or a specific sound parameter instead of the onset pattern is a desirable trait for TimeNot. A first approach to this expanded notation strategy would be to differentiate the `x` from the `X` to enable two different instruments to be attacked in the same voice.
- It is necessary to enable variables and develop further layers of abstraction. In order to automatise the production of dense canons with minimal typing TimeNot should support its algorithmic production. In order to achieve this, notation of functions should be supported as well. In a second version of this notation, it will be a priority to advance in this direction.
- The canonic structures developed by Conlon Nancarrow include divergence canons, acceleration canons, multicanonic structures. All of these should be integrated into the notation so it becomes possible to continue to explore the poly-temporality proposed by the Mexican artist. Divergence canons were implemented in the earlier Nanc-in-a-Can project, and can be easily translated into this project. Acceleration canons represent a major expansion on this software. To create a notation capable of representing acceleration canons is a major area for

exploration and imagination that could expand the current understanding of poly-temporality as posed by Nancarrow.

In this work I have tackled some questions regarding time, sound art and music in its algorithmic form guided by live coding and as well as by the topological complexity explored by Conlon Nancarrow. The dialogue and resistance foregrounded in this paper and implied in TimeNot offer a conception of time that broadens both the conceptions of time in live coding and the media/culture specificity of the music of Conlon Nancarrow. A time-oriented live-coded artistic practice can be foreseen beyond musical vanguardism and live coding technological experimentation, able to produce its own culture around time that require and invite further explorations and that pose challenging and stimulating questions. The present project lets us imagine a time that sounds less like a high-speed train running nowhere and more like an intricate and diverse ecology in the margins of sonic imagination.

URL of software repository:

https://github.com/AFrancoB/timeNot

References

Adam, Barbara. 2004. Memory of futures. KronoScope, 4, PP. 297-315.

Arom, S., & Ligeti, G. 1991. African Polyphony and Polyrhythm: Musical Structure and Methodology (M. Thom, B. Tuckett, & R. Boyd, Trans.). Cambridge: Cambridge University Press. doi:10.1017/CBO9780511518317

A.F. Blackwell, C. Britton, A. Cox, T.R.G. Green, C. Gurr,G. Kadoda, M.S. Kutar, M. Loomes, C.L. Nehaniv, M. Petre, C. Roast, C. Roe, A. Wong, R.M. Young. 2001. Cognitive Dimensions of Notations:

Design Tools for Cognitive Technology . Proceedings of the 4th International Conference on Cognitive Technology. Coventry, UK.

Bell, Renick. 2011. An Interface for Realtime Music Using Interpreted Haskell. In Proceedings of LAC 2011. Maynooth, Ireland

Bergson, Henri. 2002. Concerning the nature of Time in Henri Bergson: Key Writings. Edited by Ansell Pearson and John Mullarkey, London: Continuum.

Boym, Svetlana. 2001. The Future of Nostalgia. New York: Basic Books.

Collins, Nick. 2003. Microtonal Tempo Canons After Nancarrow/Jaffe. Proceedings of the International Computer Music Conference, Singapore.

Concheiro, Luciano. 2016. Contra el Tiempo, Filosofía práctica del instante. Mexico City: Anagrama.

Cowell, Henry. 1917.- (1996). 'New musical resources'. New York: Cambridge University Press.

Davis, Eric. 2001. "ROOTS AND WIRES": POLYRHYTHMIC CYBERSPACE AND THE BLACK ELECTRONIC. http://www.levity.com/figment/cyberconf.html. Last access: 27th February, 2018.

Deleuze, Gilles; Guattari, Félix. 1987. A Thousand Plateaus. Minneapolis: University of Minnesota Press. ISBN 978-0-8166-1402-8.

Deleuze, Gilles, Guattari, Felix. 1983. Anti-Oedipus: capitalism and schizophrenia. Minneapolis: University of Minnesota Press

Faramelli, Anthony. 2014. Resistance, Revolution and Fascism: Zapatismo and Assemblage Politics. London: Kingston University Press.

Gann, Kyle. 2006. 'The Music of Conlon Nancarrow', (New York: Cambridge University Press)

Gilroy, Paul. 1956-. (1993). The black Atlantic:modernity and double consciousness. Cambridge, Mass: Harvard University Press,

Harkins, James. 2009. A Practical Guide to Patterns.

http://distractionandnonsense.com/sc/A_Practical_Guide_to_Patterns.pdf Last accessed: 12th August, 2019.

Ikoniadou, Eleni. 2014. The Rhythmic Event: Art, Media, and the Sonic . MIT Press.

Kapur, Ajay & Cook, Perry & Salazar, Spencer & Wang, Ge. 2013. Programming for Musicians and Digital Artists: Creating Music with ChucK.

Kirkbride, Ryan. 2016. Programming in Time: New Implications for Temporality in Live Coding. In Proceedings of the 2nd International Conference on Live Coding.

Laing, Dave. 2017. An introduction to reflex. https://blog.qfpl.io/posts/reflex/basics/introduction/ Last accessed: 31 July, 2019.

Lefebvre, Henri. 2004. Rhythmanalysis: Space, Time and Everyday Life. Continuum.

Lipovaca, Miran. 2011. Learn You a Haskell for Great Good! No starch press. ISBN-13: 978-1-59327-283-8

Magnusson, Thor. 2011. ixi lang: a SuperCollider parasite for live coding. In: International Computer Music Conference, 31 July – 5 August 2011, University of Huddersfield, UK

Marcos. 2001. Our Word Is Our Weapon. New York: Seven Stories Press.

McCartney, James. 2002. “Rethinking the Computer Music Language: SuperCollider.” Computer Music Journal 26 (4). MIT Press: 61–68.

McLean, Alex, and Geraint Wiggins. 2010. Tidal–pattern Language for the Live Coding of Music. In Proceedings of the 7th Sound and Music Computing Conference.

Murcott, Dominic. 2014. Tomorrow's Music on Yesterday's Machines: In Search of an "Authentic" Nancarrow Performance. Society for Music Theory . Volume 20 Number 1.

Ogborn, David; Beverley, Jamie; Navarro Del Ángel. Luis; Tsabary, Eldad; McLean Alex. 2017. Estuary: Browser-based Collaborative Projectional Live Coding of Musical Patterns. International Conference on Live Coding (ICLC) 2017.

Paetzold, Cordula. 2010. 'Carceri d'Invenzione of Brian Ferneyhough, an analysis and composition technique', Sinefonia 14

Parr, Terence. 2007. The Definitive ANTLR Reference: Building Domain-Specific Languages. ISBN 978-0-9787392-5-6

Sorensen, Andrew, and Henry Gardner. 2010. Programming with Time: Cyber-Physical Programming with Impromptu.

Thomas, Margaret. 1999. Nancarroẃs Temporal Dissonance. Intégral 13.

TOPLAP. 2010. "ManifestoDraft." https://toplap.org/wiki/ManifestoDraft.

Toussaint, Godfried. 2003. The Euclidean Algorithm Generates Traditional Musical Rhythms. School of Computer Science, McGill University Montréal, Quebec, Canada.

Valencia, Guadalupe. 2010. Tiempos Mexicanos. Sequitur.

Villaseñor, Diego & Franco, Alejandro. 2019. Nanc-in-a-Can Canon Generator. SuperCollider code capable of generating and visualizing temporal canons critically and algorithmically. Proceedings of the International Conference of Live Coding, Madrid.

Wajcman, Judy. 2017. Esclavos del tiempo: Vidas aceleradas en la era del capitalismo digital. Grupo Planeta.

Wenzel, Jennifer. 2006. Remembering the Past's Future: Anti-Imperialist Nostalgia and Some Versions of the Third World. Cultural Critique, (62), 1-32. Retrieved from http://www.jstor.org/stable/4489234

Wessel, David; Wright, Matthew. 2002. Problems and Prospects for Intimate Musical Control of Computers, Computer Music Journal, Volume 26, Issue 3, p.11-22, (2002)

Wilson Scott; Cottle, David and Collins, Nick. 2011. *The Supercollider Book*. The MIT Press.