Digital Expression and Representation of Rhythm

Nathan Renney
Computer Science Research Centre (CSRC)
Department of Computer Science and Creative Technology
University of West of England, Bristol UK
nathan.renney@uwe.ac.uk

Benedict R. Gaster
Computer Science Research Centre (CSRC)
Department of Computer Science and Creative Technology
University of West of England, Bristol UK
benedict.gaster@uwe.ac.uk

ABSTRACT

Music provides a means to explore time by sequencing musical events in a seemingly endless and expressive way. This potential often far exceeds the ability of digital systems to enable composers and performers to explore musical time, perhaps due to the influence of Western music on implementation or maybe due to the challenges involved in the notation of music itself. In this paper we look at ways to explore time within a musical context, looking to create tangible examples and methods for exploring complex rhythmic relationships using digital systems. We draw on the approach for describing sequences in terms of cycles, inspired by the live coding language Tidal Cycles.

A simple Domain Specific Language (DSL) is described, in order to realize a Digital Musical Instrument (DMI) that facilitates performing with polyrhythm in a intuitive and tactile way. This highlights the use of DSLs for the design of DMIs. Further, an abstraction for representing sequences of musical events on a digital system is provided, which facilitates complex rhythmic relationships (namely, polyrhythm and polymeter) and extends to handle modulation of time itself.

CCS CONCEPTS

• **Applied computing** → **Sound and music computing**; • **Software and its engineering** → **Functional languages**.

KEYWORDS

Sequencing, Rhythm, Music, Time, Polyrhythm, Polymeter, DSL, DMI

1 INTRODUCTION

Rhythm is the sequencing of musical events in time. These events are set apart by a time step dictated by our sensory threshold to synchronize with these events. Repp suggests this interonset interval is between 100-120ms [16]. Variations around this duration contribute ornaments and nuanced variation, such as grace notes and the fluctuation in meter commonly referred to as 'feel' [14]. Western music theory describes music at a rate of beats per minute (BPM), as an isochronous sequence with an equally spaced, periodic pulse [5][1]. Beyond the periodic groupings referred to as meter, the emergence of pulse, the feeling of strong and weaker beats, is implied within the time signature of a musical piece and as such leans towards certain, almost idiomatic approaches to notating rhythm in traditional Western notation. Whilst traditional notation is expressive and information dense, looking towards more complex manipulations of musical time, digital systems tend to struggle to manage and represent this well. Most notably, this is a challenge in a number of software notation programs, where expressing two time signatures in parallel (polymeter) or working with concurrent time (polyrhythm) is not generally well supported and so requires undesirable 'work-arounds' in order to express these ideas, hindering the creative process.

In this paper, we build upon the notion of cyclic time as described by Tagg [18], looking at alternative ways cycles can be expressed, drawing inspiration from Tidal Cycles [12], a DSL for Live Coding of musical patterns.

There are many examples of DSLs used in music creation and performance, see for example Magnusson [11], and for the processing of audio, e.g. PD [15] and Faust [13]. However, to date these DSLs have not been widely applied to the design of Digital Musical Instruments, particularly in the context of the complete instrument, which Magnusson calls the Instrument Model [10]. Expanding on Magnusson's definition of a DMI, an instrument can be viewed through the lenses of different domains, each with their own nuances. The design of these instruments can, therefore, be composed over these domains, avoiding being tied down with or hindered by implementation details.

In this paper we introduce a small functional Domain Specific Language for manipulating musical patterns and sequences. This use of a DSL for musical patterns provides the building blocks for working with sequence based musical constructs, encapsulated in cycles, while retaining a high level of abstraction. We go on to provide abstractions that apply polyrhythmic and polymetric rhythms to our representation of time.

[1]Though there are examples of non isochronous music, which can be identified by the clapping test, described by Arom [2].

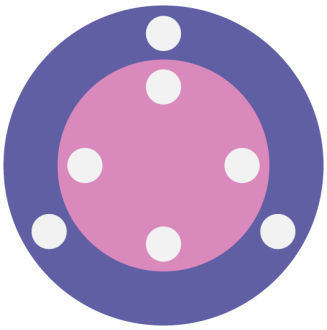


Figure 1: A 3 against 4 polyrhythm with cycles represented with circles.

To demonstrate the practical benefits of our system, we describe a DMI designed for the exploration of polyrhythm, utilizing tools from the Muses Project[2]. The DSL, pat, described in Section 3, is used to create a tangible, interactive instrument that applies an approach similar to Varney's 'Wheel Method'[3] to spread polyrhythms around a circle. This allows polyrhythms to be expressed using physical rings, as seen in Figure 1.

We conclude this introduction with a brief summary of the remaining sections of this paper:

- Section 2 Briefly demonstrates the music theory surrounding polyrhythm and polymeter, discussing motivations for facilitating more expressivity for composers when working with relatively complex rhythmic ideas on digital systems;
- Section 3 introduces a notation for describing musical patterns as influenced by Tidal Cycles;
- Section 4 provides our abstraction for representing sequences as well as manipulations that capture polyrhythm and polymeter;
- Section 5 introduces pat a small DSL for expressing patterns;
- Section 6 provides a brief overview of two instruments designed using pat and implemented to provide concrete examples of the ideas discussed in this paper; and
- Finally, Section 7 concludes with pointers to future work.

More details about the project, our implementation, and examples are publicly available from https://muses-dmi.github.io/.

[2]https://muses-dmi.github.io/pat/overview/
[3]https://ed.ted.com/lessons/a-different-way-to-visualize-rhythm-john-varney

2 TRADITIONAL EXPRESSION OF RHYTHM

Whilst comprehensively covering the music theory relating to rhythm is beyond the scope of this paper, we briefly introduce and illustrate the topics we will focus on in order to demonstrate the incorporation of these concepts, which are often difficult to express in computer notation packages and digital instruments.

Musical notation is an information dense medium that lends itself well to transferring musical information to a performer. Whilst superficially a mathematical equivalence can be shown between two methods for notating a rhythm, the choice of notation contains inferred information concerning meter and pulse for a performer to act upon.

As such it may be useful to notate work using concurrent notions of time, to cleanly portray the intended pulse of the piece. This can be viewed from two perspectives; ideally in one respect; allowing a composer to effectively express how a rhythm should 'feel' (where the pulses/strong beats of different instrumental parts should be and how to they line up against other parts) to the performers in order to fully realize the piece. Alternatively, in some performance situations, the composers realization may be compromised in order to notate a piece in a way that reads more idiomatically, allowing players that may be sight reading to read a more familiar part.

The result of these differences are subtle, constituting minor fluctuations from the strict subdivision that is notated or articulations of a musical event based on the position within the bar, but we argue for the former, which facilitates the expression of intention.

In order to avoid overly quantized playback of sequences, a digital system requires an implementation that balances a representation that is comprehensible, with a playback system that is capable of applying the more nuanced variations eluded to in this paper. As an important facet of performance, an abstraction that allows expression over this is desirable and factors into the design considerations of the approach presented in this paper, though deeper consideration of rhythmic feel and microtiming will be deferred to future work.

Beyond the musical feel embedded in a rhythm, there are also other concepts that are typically not well presented to users of digital systems. Both polyrhythm and polymetric rhythms are concepts that are incorporated in many specialized devices and applications, however, many mainstream digital systems do not typically facilitate their use in an expressive idiom.

In the following sections, these concepts are presented using short meters, allowing the concepts to be clearly displayed and fit well on the page, however, these principles extend for other values.

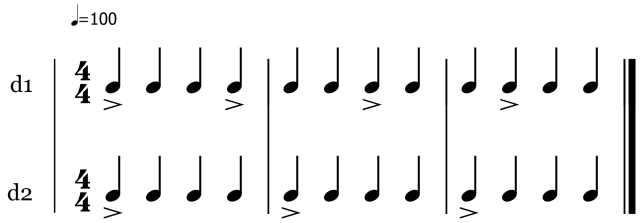


Figure 2: Polymetric passage of 3/4 over 4/4, notated in 4/4 with accents indicating the first beat.

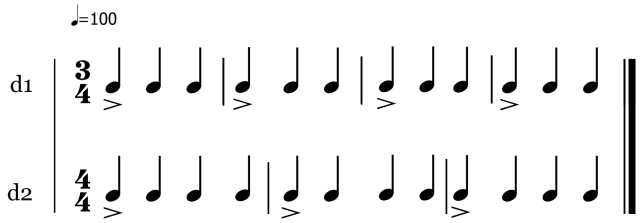


Figure 3: Polymetric passage of 3/4 over 4/4, notated in with dual time signature.

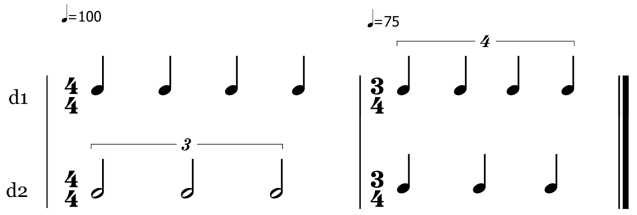


Figure 4: A polyrhythm of 4 against 3,demonstrating a notational equivalence

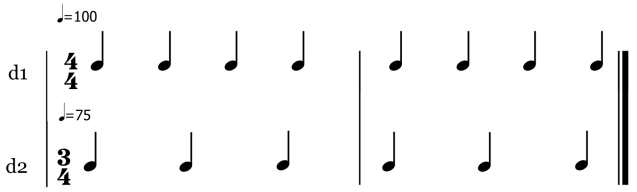


Figure 5: 4 against 3 polyrhythm simplified to quarter notes at related tempo and time signature.

Polymetric Rhythms

Consider a simple example that notates a polymetric phrase. Phrases are constructed using note durations rooted in the same tempo, but using different rhythmic meters (time signatures). This concept is often expressed by notating the parts in a common time signature, perhaps providing articulations that are suggestive of the 'feel' of a different meter. This is demonstrated in Figure 2, where the first beat of a phrase is accented.

In some cases, a composer may wish to work directly in the time signatures that the work was intended to be in, producing a notation similar to that shown in Figure 3. This allows idiomatic writing for both parts and implies the feel of each part individually. Despite, fairly common use, this is not supported by many mainstream digital systems, and often when it is, there are restrictions that prevent a complete sense of expressibility.

Polyrhythmic Rhythms

Figure 4, shows two approaches to expressing a polyrhythm. In the first measure,a four over three polyrhythm is expressed as a 4/4 measure, at a tempo of 100BPM. This is demonstrated as being equivalent to the second measure, notated in a different meter and tempo.

Using traditional notation, these phrases are expressed quite well indicating how these rhythms anchor against each other. In more complex examples, however, it may be beneficial to conceptually work with tempo and time signatures applied against each other rather than having some parts entirely in tuplets[4]. This is particularly, the case where polyrhythms form the basis for an entire piece, and further, the intention may be to apply both different tempo and pulse. Figure 5 demonstrates how this may be notated, where the implied pulse from each time signature should be considered.

Rate

In Western notation, the rate at which a rhythm is performed is represented in a relative manner, with durations for notes given as subdivisions of a measure of time (a bar). This measurement is made absolute by providing the number of beats per minute (BPM). While BPM allows a performer to approximate tempo in relation to seconds, the approach described in Section 4 assumes rate to be given as the duration of the sequence. It is therefore important to consider this relationship where duration (in seconds) of a measure can be calculated from beats per minute (BPM) with the following equation, with B representing the number of beats in the measure.

$$\frac{60}{BPM} \times B$$

As mentioned previously, traditional notation leaves perturbations in tempo to be inferred based on the notation (time signature, articulation and style), but it is recognized

[4]In practise this may require musicians to perform using individual metronomes, in order to realize the intention effectively.

that variation of the interonset interval between notes also significantly contributes to the 'feel' [6]. This is typically contextual, with proficient performers modulating the tempo based on the style of piece being performed.

Limitations in Expression

Many electronic instruments and software do facilitate expression of these rhythmic ideas, however, it is observable that there are often hurdles in expressing complex rhythmic relationships that require working against the functionality of the application or device. Further, a common complaint levied against digital systems is the lack of 'feel'. This is due to many implementations notions of metre and pulse lacking the continuous modulation added by a human performer. While there are systems that aim to capture this, we consider there a lack of work that abstracts these ideas in a way that is transferable and expressive. This therefore motivates the approach laid out in this paper.

These issues are beginning to be addressed in different systems, but a consolidated approach that affords expressivity to the composer or performer is missing.

3 DESCRIBING TIME WITH TIDAL INFLUENCED PATTERNS

Programming languages, particularly those from the live coding movement [4] offer another method for exploring sequences, with a focus on sequencing being a staple part of many 'live coding languages' [1, 11]. Within these languages, code is used to express a sequence during performance. The ideas presented in this paper draw from and are influenced by these languages, later drawing on them within the context Digital Musical Instrument design.

Whilst there are a number of programming languages that have been built to express musical ideas, Tidal Cycles captures rhythmic expression in a way that is transparent when working with rhythmic sequences and is syntactically light.

Due to the density of musical information that traditional notation presents, it is difficult to provide a rich and expressive representation of music as a text based programming language. Focusing on music based on patterns in time, Tidal Cycles excels. Sequences are expressed as cycles, analogous to bars or measures, though they do not inherently suggest a meter. A single cycle is a length of time into which some number of events may be distributed. Musical events are distributed equally throughout a cycle and any single event may be further subdivided by providing subdivisions (described as nested lists) of musical events.

For describing patterns, we derive a variation of Tidal cycles syntax, where a pattern is a string, delimited by white space. Further subdivisions are expressed using a notation for nested lists, incorporating the most fundamental ideas

Figure 6: An example pattern featuring subdivisions down to 16th notes.

of Green's cognitive dimensions for notating lists [7] [5]. This syntactically allows, for example, expressing the pattern of two sixteenth notes, one eighth note and three quarter notes as seen in Figure 6, using the following pattern[6]:

```
[[bd bd] bd] bd bd bd
```

Whilst Figure 6 notates the pattern above in 4/4 we should consider that the pattern, unlike the notation, has no implication of how the pulse of the part should feel. In the context of this work, pulse and meter may be considered functions that act on a pattern and as such are not represented in the pattern itself which will be built upon in future work.

4 NOTIONS OF TIME

This section describes an abstraction for musical time, providing an underlying data structure that can be used by digital systems for performance and playback, termed sequences. Further, constructs for the manipulation of patterns are provided. The notation described in Section 3 is used for patterns, showing how they are translated to 'flattened' sequences.

The approach presented here is intended as a conceptual model, rather than a strict format specification. As such, it may be extended with front matter/meta data for each sequence in order to encode the articulation and phrasing that is applied to the sequence, as well as other implementation specific requirements.

A complete implementation based on the ideas presented in this section can be found on the project's Github page[7].

Representing Sequences

A sequence is a series of events, taking the form of a data structure that approximates Schaeffer's definition of a sound object [17], with a collection of functions which are able to manipulate them.

A sequence of events in time is represented as an ordered list[8], where each element represents the onset of a given set of events:

[5] We use Haskell's list notation, [] for empty lists and [x1 ,..., xn] for lists containing n elements, where x_i could also be a list.

[6] Technically this should have been written "[[bd bd] bd] bd bd bd", but we emit the quotes when clear from context.

[7] https://github.com/muses-dmi/pat/

[8] As already noted we use Haskell's list notation, [x1 ,..., xn], to represent an ordered set.

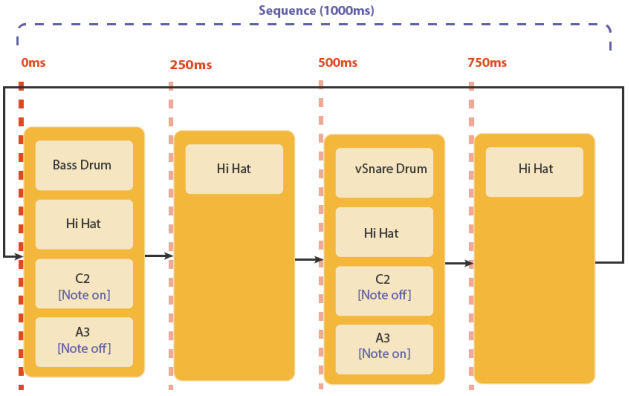


Figure 7: Sequence structure

$$E = [e_1, e_2, \ldots, e_n]$$

Events at a given point in time happen simultaneously and, therefore, do not require ordering.

Specific representations of events are undefined and left to a particular implementation, examples include MIDI[9] or OSC[10] messages. This structure, called a sequence, is assigned the following type, where τ is the type for events and is supplied by an implementation[11]:

$$\texttt{sequence} : [[\tau]]$$

The position of events in the list represent events in time, with the gaps between considered the interonset interval, analogous to the interval explored by Madison, for the perception around intertap interval [9]. The sequence can be naively played by stepping between each element of the list with a fixed time interval. Modulation of this playback interval remains an exciting opportunity for future work.

In order for a sequence to be played, the interonset interval must be supplied as a function of the patterns used to generate it. Therefore, an implementation requires a sequence and an interonset interval, derived from some notion of rate (cycle duration or tempo) in order to operate. Figure 7, provides a visual representation this[12].

Tempo calculations

Devices utilizing this representation are required to calculate and manage tempo. Separating the management of interonset intervals from the sequence of events, such that the tempo is free to be modulated without the need to operate

[9] https://www.midi.org/
[10] http://opensoundcontrol.org/
[11] A expression of the form $x : \tau$, states that x has type τ.
[12] Observe that if MIDI or a similar mechanism is used and a sequence is being used as a loop, the note off will be required to be on the starting note of the 'next conceptual cycle', meaning an unpaired note off will be sent on the first beat of the first playback of the cycle, as seen in this case.

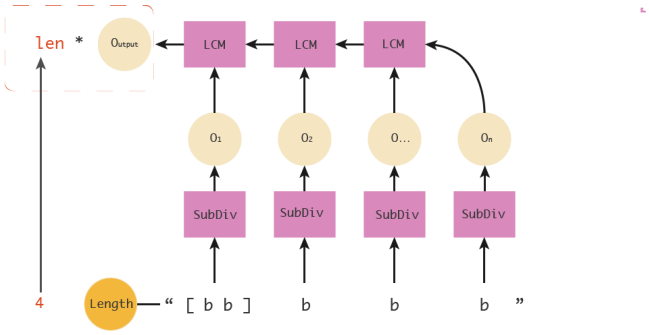


Figure 8: The process of expanding a pattern's subdivisions.

on the sequence itself. This provides a opportunity to reflect real world variations in time such as those described by Barton [3].

Given a target cycle duration, the interonset interval can be calculated as follows, where T is the length of time a cycle lasts and N is the number of cycles. These values are then divided by the total number of steps in the sequence to provide the interonset interval, defined as follows:

$$\texttt{interval} = \frac{T \times N}{Steps}$$

Translating Patterns to Sequences

As described above patterns form an ordered list, where sublists represent nested subdivisions. Non list elements, i.e. elements that are not subdivisions, represent musical events in time.

Translating patterns to sequences, with explicit quantization, is the process of 'flattening' a pattern such that it contains no subdivisions and is correctly spaced with respect to time. This process is straightforward in the case of a single pattern, but requires a different approach for polyrhythmic and/or polymetric composition of multiple patterns. We first consider the case of 'flattening' a single pattern to a sequence and then use this to account for multiple patterns, including polyrhythmic and/or polymetric composition.

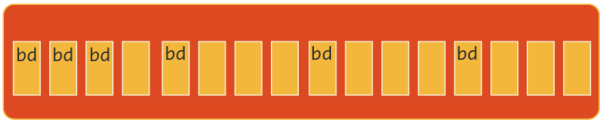
Flattening Subdivisions

In order to create the ordered list of events that represents a sequence the subdivisions of the input patterns must be 'flattened', such that each step between elements represents the smallest possible interonset interval that can represent the list.

Consider as an example the notation from Figure 6, which is described by the following pattern:

```
[[bd bd] bd] bd bd bd
```

The smallest subdivision used is a sixteenth note, which implies that the sequence must be flattened to the following:



The expansion of subdivisions is a function from a pattern to a sequence (i.e. pattern → sequence). This translation results in a sequence that has a length equal to the least common multiple of every subdivision in a pattern, multiplied by the length of the pattern itself. This operation is demonstrated over a pattern in Figure 8.

Handling Polyrhythmic and Polymetric

We now consider the creation of sequences that combine patterns in either a polyrhythmic or polymetric manner, utilizing the previously discussed method for expanding subdivisions.

Polyrhythmic Merge

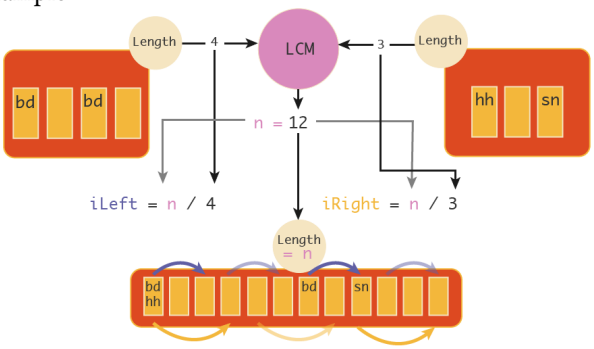
Polyrhythmic merge is the process of first expanding the subdivisions for both sequences and then creating a new list that is n elements in length, where n is the least common multiple of the 'flattened' pattern's length, e.g.:

```
lcm (flatten l) (flatten r)
```

where l and r are short for left and right, respectively. Each element is then inserted into the new list, at intervals of i, calculated for left and right independently as:

```
iLeft = n / (length (flatten l))
iRight = n / (length (flatten r))
```

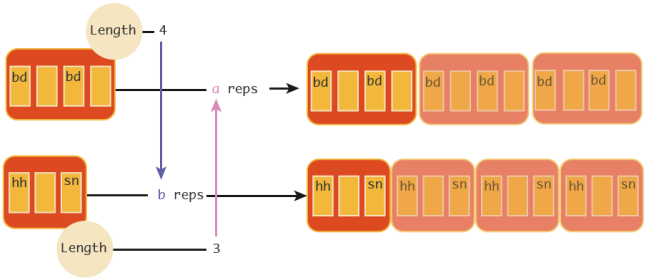
The process is shown diagrammatically in the following example:



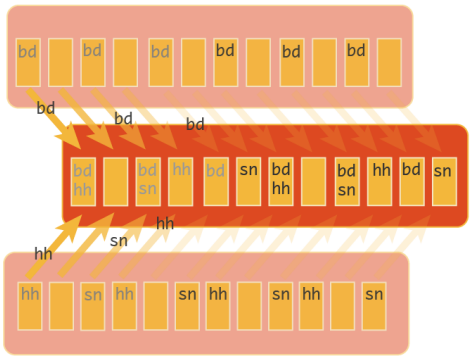
Polymetric Merge

The process of merging a polymetric phrase creates a new sequence that extends to the point that the input sequences synchronize. This is calculated by multiplying the length of both sequences prior to flattening, which is achieved by repeating the list.

This is demonstrated in the following diagram:



The resulting sequences are then combined such that each element becomes a list of lists, where the nested list contains the events on that subdivision, as demonstrated in the following diagram:



Patterns can then be be subdivided as described previously.

5 AN EXAMPLE DSL FOR EXPRESSING NOTIONS OF TIME

To demonstrate the use of our representation this section explores using DSLs (as described by Hudak [8]) in the context of Digital Musical Instrument Design. A tiny DSL, called pat, for creating patterns in time' with a syntax inspired by Tidal Cycles is presented, whose grammar is specified in Figure 9. We assume a symbolic representation for events, e.g. alpha numeric sequences such as bd and snare, etc. The binary operations |:| and −:− represent polyrhythmic and polymetric merge, respectively.

pat can express sequences with a light touch. For example, consider the following, terse, description producing a 3 : 4 : 7 polyrhythmic pattern:

```
[a c e] a a |:| a c a c |:| g g g g g g g
```

Further, these combinators can be used together to create complex, evolving patterns. The following expression generates twenty four steps of evolving musical material[13]:

```
[a c e] a a [a c e] a a
−|− a c a c
|:| g [g c] g [g c] g [g c] g
```

[13]This example can be heard on https://muses-dmi.github.io/pat/listen

⟨*event*⟩ ::= identifier

⟨*pattern*⟩ ::= ⟨*event*⟩
| [⟨*pattern*⟩ { , ⟨*pattern*⟩ }]
| ⟨*pattern*⟩ |:| ⟨*pattern*⟩
| ⟨*pattern*⟩ -:- ⟨*pattern*⟩

Figure 9: Pattern grammar

Figure 10: Screen shot of Muses Audio Application.

A practical realization, as given on the project's Github and outlined in Section 6, must provide concrete representations for events and other implementation details omitted here, for ease of presentation.

6 APPLICATIONS FOR DMI

Given the DSL pat, we now briefly discuss how it might be utilized in the design of a novel instrument that facilitates the ability to both describe constraints (in the form of rhythmic sequences) and also the ability to perform. We provide an example of two instruments, one virtual and one physical.

The Muses Synth and Pat

A combination of the Muses Synth[14] and pat can be used to describe a virtual instrument.

In order to operate with the Muses Synth we transpile from our representation into a JSON file representing a sequence. The Muses Synth implements the ability to parse and perform a sequence as described in Section 4. This provides the ability to define a rhythmically complex Virtual DMI using pat expressions. This instrument can be downloaded and explored at the project's website.

[14] https://muses-dmi.github.io/pat/synth

Figure 11: A novel instrument for polyrhythmic expression with marbles.

Polyrhythmic Ring Sequencer

This instrument provides exploration of polyrhythm that features desirable qualities for a pedagogic DMI. Namely:

- Accessible - requiring little technical proficiency
- Playable - can be played in real time, constituting a performance that requires input from a performer.
- Visually represents polyrhythm in an intuitive manner.

The instrument produced for this purpose is shown in Figure 11. This instrument is a circular step sequencer with three channels, represented by three concentric rings. It is implemented using a Sensel Morph[15], a touch pad capable of measuring force, and uses marbles to represent musical events. A cycle is represented from 12 o'clock, around 360°.

pat is used for the description of the rings with a set of functions that use ImplicitCAD[16] to produce an STL, a 3D model format for 3D printing. We allow the | : | combinator to be used but restrict the use of polymetric merge (−|−) due to limitations of the system. This implementation is provided on project the website, alongside tools to generate new rings based on pat.

7 CONCLUSION

This paper presented a method for representing musical sequences as an ordered set, containing musical events. An event may be considered analogous to Schaeffer's definition of a sound object [17], where some representation describing a musical event is provided.

This method is motivated by a desire to be able to manipulate the sequence to represent complex notions of time such as polyrhythm and polymeter. A process for merging sequences inline with these concepts is provided, producing

[15] https://sensel.com/pages/the-sensel-morph
[16] http://www.implicitcad.org/

a sequence of events that represents merging the provided sequences.

A conceptual overview of how sequences may be merged both polyrhythmically and polymetrically is demonstrated as alongside how time is represented within this context. This is presented at a high level in anticipation of further work, where this method produces a suitable digital representation to explore more complex and expressive manipulations of time.

In order to explore this concept, a small language was presented to express patterns in time, inspired by the language Tidal Cycles [11]. This facilitates the expression of a polyrhythmic pattern of seven snare hits over a 'four on the floor' bass drum pattern, in the following manner:

```
bd bd bd bd |:| sn sn sn sn sn sn sn
```

To provide real world examples, and to position future work on the topic, two digital musical instruments were introduced, that incorporate the use of our pattern language in their design. These instruments are made available for further exploration of how this overall approach may be implemented in a digital system. Additionally they provided a foundation for future work, where we intend to develop a variety of small Domain Specific Languages in the design of Digital Musical Instruments, to create abstractions that allow for expressivity in the design phase, without requiring concern for implementation.

REFERENCES

[1] Samuel Aaron and Alan F. Blackwell. 2013. *From Sonic Pi to overtone: Creative musical experiences with domain-specific and functional languages*. Vol. 12. Taylor & Francis. 35–46 pages. https://doi.org/10.1145/2505341.2505346

[2] Simha Arom. 2004. *African polyphony and polyrhythm: musical structure and methodology*. Cambridge university press.

[3] Scott Barton, Laura Getz, and Michael Kubovy. 2017. Systematic Variation in Rhythm Production as Tempo Changes. *Music Perception: An Interdisciplinary Journal* 34, 3 (2 2017), 303–312. https://doi.org/10.1525/mp.2017.34.3.303

[4] Nick Collins, Alex McLean, Julian Rohrhuber, and Adrian Ward. 2003. Live coding in laptop performance. *Organised Sound* 8, 3 (2003), 321–330. https://doi.org/10.1017/s135577180300030x

[5] W. Tecumseh Fitch. 2013. Rhythmic cognition in humans and animals: distinguishing meter and pulse perception. *Frontiers in Systems Neuroscience* 7 (10 2013), 68. https://doi.org/10.3389/fnsys.2013.00068

[6] Fabien Gouyon. 2007. Microtiming in "Samba de Roda" Preliminary experiments with polyphonic audio. *Brazilian Symposium on Computer Music* January 2007 (1 2007).

[7] T R G Green. 1989. Cognitive Dimensions of Notations. In *Proceedings of the Fifth Conference of the British Computer Society, Human-Computer Interaction Specialist Group on People and Computers V*. Cambridge University Press, New York, NY, USA, 443–460. http://dl.acm.org/citation.cfm?id=92968.93015

[8] Paul Hudak. 1996. Building domain-specific embedded languages. *Comput. Surveys* 28, 4es (12 1996), 196–es. https://doi.org/10.1145/242224.242477

[9] Guy Madison. 2001. Variability in isochronous tapping: Higher order dependencies as a function of intertap interval. *Journal of Experimental Psychology: Human Perception and Performance* 27, 2 (4 2001), 411–422. https://doi.org/10.1037/0096-1523.27.2.411

[10] Thor Magnusson. 2010. Designing Constraints: Composing and Performing with Digital Musical Systems. *Computer Music Journal* 34, 4 (12 2010), 62–73. https://doi.org/10.1162/COMJ{_}a{_}00026

[11] Thor Magnusson and Alex McLean. 2018. Chapter 14: Performing with Patterns of Time. *Oxford University Press* (2 2018), 1–17. https://doi.org/10.5281/ZENODO.1193251

[12] Alex McLean. 2014. Making programming languages to dance to. *Proceedings of the 2nd ACM SIGPLAN international workshop on Functional art, music, modeling & design - FARM '14* (2014), 63–70. https://doi.org/10.1145/2633638.2633647

[13] Yann Orlarey, Dominique Fober, and Stephane Letz. 2009. FAUST: An Efficient Functional Approach to DSP Programming. *New Computational Paradigms for Music* 290, May 2015 (6 2009), 65–97.

[14] Rainer Polak. 2010. Rhythmic Feel as Meter: Non-Isochronous Beat Subdivision in Jembe Music from Mali. *Society for Music Theory* 16, 4 (12 2010), 1–26. http://www.mtosmt.org/issues/mto.10.16.4/mto.10.16.4.polak.html

[15] Miller Puckette. 1996. Pure Data. In *Proceedings of the 1996 International Computer Music Conference*. International Computer Music Association, San Francisco, 269–272.

[16] Bruno H Repp. 2003. Rate Limits in sensorimotor Synchronization With Auditory and Visual Sequences. *Journal of Motor Behavior* 35, 4 (2003), 16.

[17] Pierre Schaeffer. 2017. *Treatise on musical objects : an essay across disciplines*. Vol. 20. Univ of California Press. 569 pages.

[18] P Tagg. 1997. Understanding Musical Time Sense: Concepts, Sketches and Consequences. http://tagg.org/articles/xpdfs/timesens.pdfwww.tagg.org