NaN (Not-a-Notation) - can petri-nets be utilized as an interactive notation for live coding?

Christopher Taudt (harte echtzeit)
futur AAA
info@harte-echtzeit.com

ABSTRACT

Usually, the "liveness" in live coding performances is either done by playing from scratch or by utilizing some prepared parts/phrases which are used by the artists to alter or improvise on. Very rarely, input from audience is included. This work will explore if and how petri-nets can be utilized as an interface for external input. It will investigate how a meta-language based on petri nets can help to control the flow of a live coding performance, alter its structure and patterns, visualize it and more abstractly serve as a notation. It will be analyzed how it can aid an artist in writing a notation for tracks while also maintaining flexibility for external inputs like an audience, situations and instruments. After an introduction to petri nets, a simple example for an implementation in tidal cycles is given. The opportunities for such an approach as well as development possibilities for other live coding languages are discussed. This paper submission is accompanied by a proposal for a performance to test the approaches in a live situation.

1 Motivation and scope

Live coding has evolved as an artistic praxis drastically in terms of approaches, technologies, genres and styles. Apart from technical detail, the increasing number of users has also brought an increasing number of approaches to perform in an actual live situation. However, most performances can be catagorized into two techniques: prepared code and from-scratch. Here, pure from-scratch sessions are similar to improvisation practices in other forms of musical performance which are driven by the artists experience and attention in the moment. Performances based on prepared parts are often used to improvise on while the prepared code comments are used as a very raw form of notation. Rarely, the audience is included in performances by utilizing different interfaces.

1.1 Notation systems status-quo

In the world of musical performance, playing from sheet is the de-facto standard which implies a common notation system. In western music, staff notation has developed as the quasi standard for musical representation, (*The Cambridge History of Western Music Theory* 2002). During the developemnt of more experimental genres of modern music, notable composers such as Steve Reich, John Cage and others developed their own more graphical notation of music, (Pryer 2011). Artists such as Mark Applebaum and Hans-Christoph Steiner have developed these graphical notations even further by utilizing software such as Pure Data, (Steiner 2004). Usually most of the notation systems strive for being abstract and mostly instrument-independent (notation of pitches, rhythms, etc.), descriptive of the flow of time or a progression, delivering a visual clue of the structure and therefore visual guidance, enabling reproducibility of a musical piece as well as sometimes enabling decision making (e.g. choice or improvisation options). Live coding seems to blur the line between an instrument and a notation, as it partly implements the features mentioned above such as structure given by patterns which are agnostic to the sound played and partly omits them, e.g. as reproducibility and flow of time can be changed drastically by the artist. The author has himself worked on a very basic nottion system for his own usage, but has com e to the conclusion that this was not able to satisfy the features mentioned above, (Taudt 2023).

1.2 Scope of this work

Within this work, it will be investigated how petri-nets, which have already been discussed as a description system for musical applications (Loy 2006), can be utilized for live coding applications. Furthermore, it will be investigated if it can be abstract, time descriptive, visually helpful and enabling reprocuibility of live coded music. Alongside, the aspects how this approach can aid audience interaction and how the perti-net notation can serve as a meta-language which in itself would be interpretatble by a program are discussed briefly.

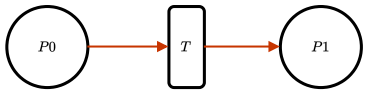

Figure 1: *A simple petri net structure where a place $P1$ will transfer its tokens due to the transition T to a second place $P2$*

2 Experimental work

2.1 Petri nets as a suggestion

Petri nets (PN) are an abstract way of representing flow of information similar but more general than a state machine, (Rozenberg and Engelfriet 1998). They are abstract and very concise, have no inherent temporal dependency but can be enhanced by it, consist of *places*, *transitions* and *arcs* where *places* hold a state and can act as a switch to enable/prevent flow, *transitions* change states in *places* and *arcs* deliver auxilliary information. They can be graphically represented as in Fig. 1. In this sense, petri-nets form very basic building blocks which can be chained together. This abstract nature allows a very flexible usage and lends itself to implement a very basic notation based on in the proposed sense.

2.2 Minimal case study - Haskell implementation

Due to their abstract nature, an implementation in a functional manner in *Haskell* is possible

```
place in_num max_num = if in_num > 0 && in_num <= max_num then True else False
trans in_state = if in_state then 1 else 0
```

Here, a place is defined as a function with an input number of tokens *in_num* and its maximum number of tokens *num_max*. Upon calling, *in_num* is evaluated and a binary value is returned. Accordingly, a transition is defined as a function with a binary *in_state* which is evaluated and an integer output which can be backfed into a place as the *in_num* input. Utilizing these basic operations, a simple petri net as decribed in Fig. 2 can be implemented

```
linP b = place (trans (place b 1)) 1
```

In this case, *linP* is parametrized with the variable *b* which is used as *in_num* while the *max_num* is fixed at 1. The binary result can be utilized to trigger musical actions such as a change in patterns

```
let iclc = "9 3 12 3"
    twentyFive = "2 0 2 5"

patChoose x = if x then iclc else twentyFive

musicalActionA pat_in =
    d1 $ fast 2
    $ note pat_in
    # s "bass3"
    # cut 1
    # gain 0.9

do
    let trig = linP 0
    musicalActionA (patChoose trig)
```

In this very basic example, the simple act of pattern switching is abstracted in the *do... musicalAction* phrase. It can be described in relation to the basic structure as in Fig. 2.

Based on this atomic structure, it can be envisioned how more complex behaviour can be modeled. One possibility is the combination of multiple linear petri nets as in *linP*

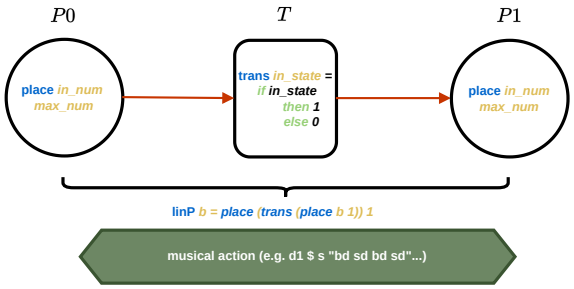

Figure 2: *A simple petri net structure where a place $P1$ will transfer its tokens due to the transition T to a second place $P2$*

```
polyP c d = (linP c, linP d)

musicalActionB trig1 trig2
 | trig1 && trig2 = do
     d2 $ s "bd sd ~ bd"
     d3 $ s "hh*3 cp"
 | trig1 = do
     d2 $ s "bd sd ~ bd"
     d3 $ silence
 | trig2 = do
     d3 $ s "hh*3 cp"
     d2 $ silence
 | otherwise = do
     d2 $ silence
     d3 $ silence

do
  let (track1, track2) = polyP 0 0
  musicalActionB track1 track2
```

In this case, the user passes two values to *polyP* which is chaining two *linP* structures. This enables the triggering of four different musical actions. One can clearly see that the possibilities of stcking and combination are huge. Furthermore, an example utilizing a random choice is given in the source file for this paper in the linked git repository.

2.3 Minimal case study - multiple tokens

In a live situation, it might be interesting prepare a temporal structure of phrases to be played and then improvise on them or tweak them interactively. In order to achieve the control of temporal and sequential behaviour using a petri net, the a second kind of transition is to be defined.

```
trList :: Int -> Bool -> [Int]
trList in_num in_state
  | in_num <= 0 = [ ]
  | otherwise = 1 : trList (in_num - 1) (p (in_num - 1) (in_num - 1))
```

In this definition, the transition emits a list of tokens which can then be processed by a resulting place into a musical action of choice. In the follwong example, the resulting place =playPiece= not only plays a prepared phrase for the number of entries resulting from =trList= but also makes a decision on which phrases should be played according to the value of the current list entry

```
import Control.Concurrent

cycTime = round(1/0.5625)
```

```
musicalActionC :: Int -> IO ()
musicalActionC x
  | x==1 = do
      once $ fast 2 $ stack[s "bd cp bd sd",
                            note iclc # s "bass3" # cut 1 # gain 0.8]
      waitHere
  | otherwise = do
      once $ fast 2 $ stack[s "hh*8",
                            note twentyFive # s "bass3" # cut 1 # gain 0.8]
      waitHere
  where waitHere = threadDelay (cycTime * 1000000)

playPiece [ ] = d1 $ silence
playPiece x
  | head x==1 = do
      musicalActionC 1
      playPiece (tail x)
  | otherwise = do
      musicalActionC 2
      playPiece (tail x)

do
  let token = 2
      times = trList token (place (token-1) token)
  playPiece times
```

It can be easily imagined how this construction can be used to either randomly change the order to the phrases or use physical buttons distributed among the audience to let them decide[1]. Chaining the *playPiece* portion and extending it with other variables will allow for an abstract and yet flexible description of the structure of a live coded piece.

3 Discussion

The implementation in *Haskell* is not ideal as it is graphically not very expressive and a bit cumbersome. In the future, it would be better to implement it in e.g. *Pure Data* or any other graphical system. Additionally, it would be interesting to return the values of the petri net as e.g. OSC messages so that multiple other live coding languages can receive the input simulataneously. This would enable interesting multi-artist performances and lead the way to greater abstraction. The authors wants to encourage a more intense discussion on the topic of notation in the live coding community. This could enable the definition of some basic building blocks and symbols which will be the vocabulary of a possible notation system. This definition must be driven by many practicing artists from different backgrounds and live coding languages. A challange when using the notation in a live setting will be not to obstruct code by the abstraction.

4 Summary and Outlook

This work showed briefly that petri-nets can act as an abstract notation system for live coding. It has been showm that they can be utilized for simple decision making and timing procedures by a basic implementation in Haskell. This work is supposed to give only an initial impulse for the topic and leaves therefore a lot of open tasks and questions. Some of them are: Is the proposed notation system capable of beeing used in live and studio situations? Can we utlize a graphic implementation to visualize the notation and control multiple live coding environments? Is the level of abstraction enough to use it flexible with different live coding languages and is it expressive enough to be digested by an audience as well as by th artists? This proposed system is still in an early stage and will be tested during the ICLC 2025 conference in a live performance. Furthermore, the author encourages the reader to work with the code, github repository.

[1] Sure enough, Tidal Cycles has build-in functions for sequencing, but the aim of the petri net formalism is to develop a metalanguage/notation which can interact with multiple live coding languages.

References

Loy, Gareth. 2006. *Musimathics, Volume 1: The Mathematical Foundations of Music*. The MIT Press. http://www.jstor.org/stable/j.ctt5hhmz4.

Pryer, Anthony. 2011. "Graphic Notation." Oxford University Press. https://doi.org/10.1093/acref/9780199579037.013.3008.

Rozenberg, Grzegorz, and Joost Engelfriet. 1998. "Elementary Net Systems." In *Lectures on Petri Nets i: Basic Models: Advances in Petri Nets*, edited by Wolfgang Reisig and Grzegorz Rozenberg, 12–121. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-65306-6_14.

Steiner, Hans-Christoph. 2004. https://at.or.at/hans/solitude/.

Taudt, Christopher. 2023. https://github.com/harte-echtzeit/not-a-notation.

The Cambridge History of Western Music Theory. 2002. The Cambridge History of Music. Cambridge University Press.