Creative Quantum Computing

Inverse FFT Sound Synthesis, Adaptive Sequencing and Musical Composition (Original Paper by Eduardo R. Miranda)

Samyak Surti

Quantum Computing at Berkeley

Spring 2021

Quantum Computer Music

Introduction

- University of Plymouth's Interdisciplinary Centre for Computer Music Research is at the forefront of this new field of research, now being referred to *Quantum Computer Music*.
- Showcased musical composition : *Zeno*

General Order of Topics

- Brief introduction to Quantum Computer Music
- Algorithmic Music Systems
- Development of *Zeno*

Quantum Computer Music

Origins and Current State of Field

- Computer Science and Computer Music, or rather computer-generated music, have been concurrently developed.

Examples

- 1940 - CSIR installed loudspeaker on Mk1 computer to track progress of a program using sound.
- 1951 - Geoff Hill, mathematician with a musical background programmed the Mk1 to play a tune.
- Late-1950s - Lejaren Hiller (Professor of Chemistry) and mathematician Leonard Isaacson at UIUC programmed the ILLIAC computer to compose a string quartet.

- Quantum Computer Music presents itself as a natural progression along the journey of music and computing.

Algorithmic Music

Intuition

- Early approaches utilized algorithms fitted to rule sets to "compose" music.
- Another approach makes use of probability distributions.

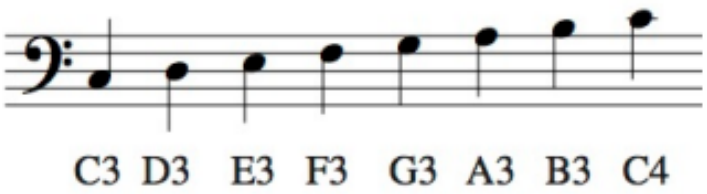


Fig. 1. A given ordered set of musical notes.

Figure: Sample note transition rule set

Rule 1: if C3, then either C3, D3, E3, G3 or C4
Rule 2: if D3, then either C3, E3 or G3
Rule 3: if E3, then either D3 or F3
Rule 4: if F3, then either C3, E3 or G3
Rule 5: if G3, then either C3, F3, G3 or A3
Rule 6: if A3, then B3
Rule 7: if B3, then C4
Rule 8: if C4, then either A3 or B3

Algorithmic Music

Intuition (Cont.)

	C3	D3	E3	F3	G3	A3	B3	C4
C3	0.2	0.2	0.2	0.0	0.2	0.0	0.0	0.2
D3	0.33	0.0	0.33	0.0	0.33	0.0	0.0	0.0
E3	0.0	0.5	0.0	0.5	0.0	0.0	0.0	0.0
F3	0.33	0.0	0.33	0.0	0.33	0.0	0.0	0.0
G3	0.25	0.0	0.0	0.25	0.25	0.25	0.0	0.0
A3	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0
B3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
C4	0.0	0.0	0.0	0.0	0.0	0.5	0.5	0.0

Fig. 2. An example of a transition matrix.

qSyn (Quantum Synthesizer)

Brief Overview

- **qSyn** - An interactive inverse Fast Fourier Transform parameterized by a quantum hyper-die (simple quantum circuit).
  1. Will first listen to a tune or melody being played.
  2. Enumerate the number of notes that were played.
  3. Synthesizes as many sounds as there were notes played.
  4. The timbres of the synthesized sounds are decided by the quantum hyper-die.
- One of the goals of the final outputted synthesized sound is that it should sound nothing like the listened melody in anyway.

Fourier and Fast Fourier Transform

Building Intuition

- In essence, the Fourier Transform takes functions in the time-domain to the frequency-domain and vice versa.
- Forward Fourier Transform - Analaysis Equation

$$X(\omega) = \int_{-\infty}^{+\infty} x(t) e^{-j\omega t} dt$$

- Inverse Fourier Transform - Synthesis Equation

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} X(\omega) e^{j\omega t} d\omega$$

- The Inverse Fast Fourier Transform is implemented in this paper.

Fourier and Fast Fourier Transform

Building Intuition (Cont.)

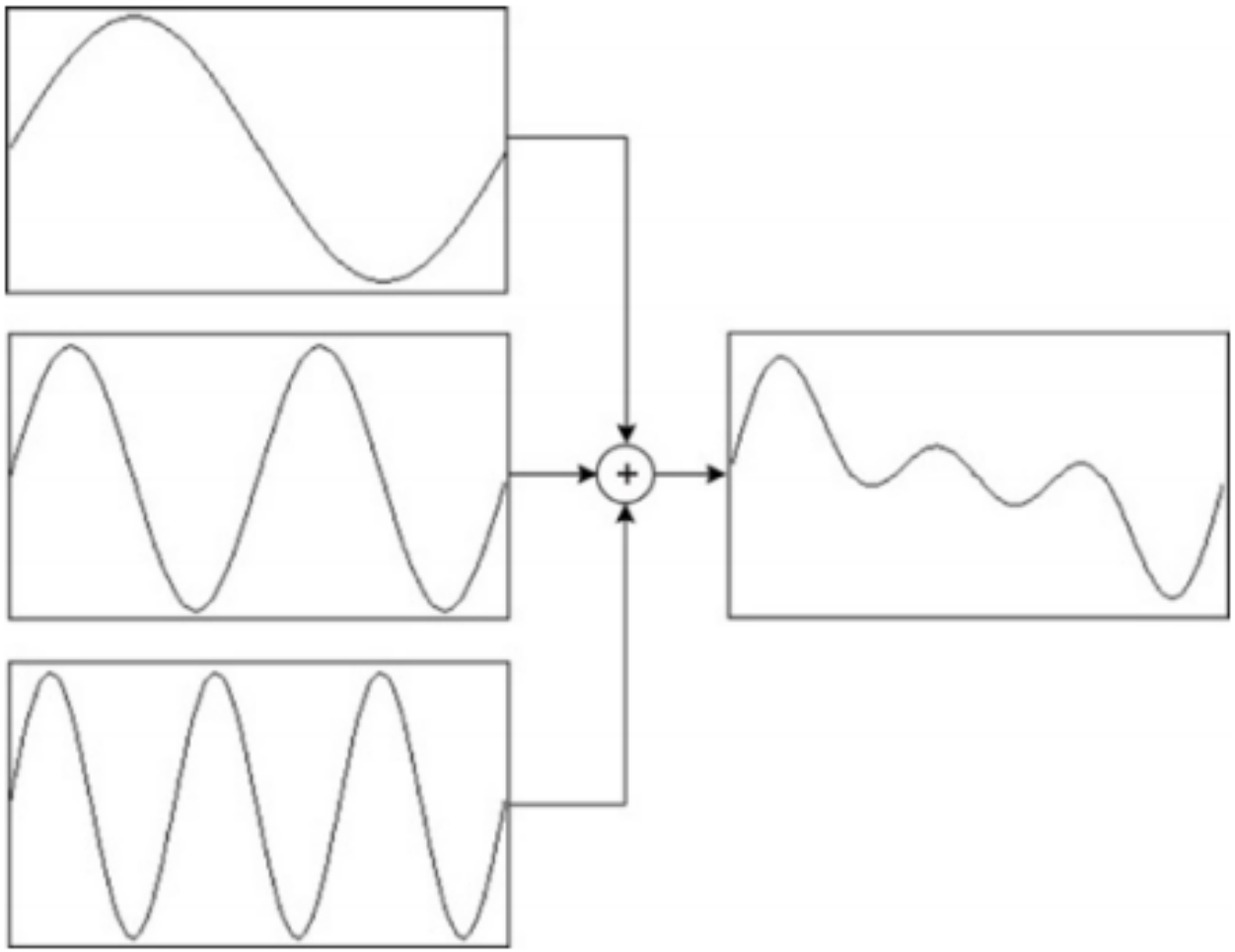


Fig. 3. Example of inverse FFT synthesis of 3 sinusoidal waves.

qSyn (Quantum Synthesizer)

Implementation

Building Blocks of qSyn

- 8 oscillators producing sinusoidal waves (notes)
- Oscillator outputs are summed $\rightarrow$ LFO (low frequency oscillator) is applied, creating vibrato effect
- ADSR (Attack, Decay, Sustain, and Release) "envelope generator" used to adjust overall output amplitude

Recall quantum hyper-die...

- Ex: Suppose we have a list of 8 frequencies : $Freq = [f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7]$. Triplet (0 1 0) corresponds to decimal number 2, thus we'd retrieve f_2.
- Hyper-die utilized two-times per sound, first time to retrieve frequences, next time to retrieve amplitudes

qSyn (Quantum Synthesizer)

Implementation (cont.)

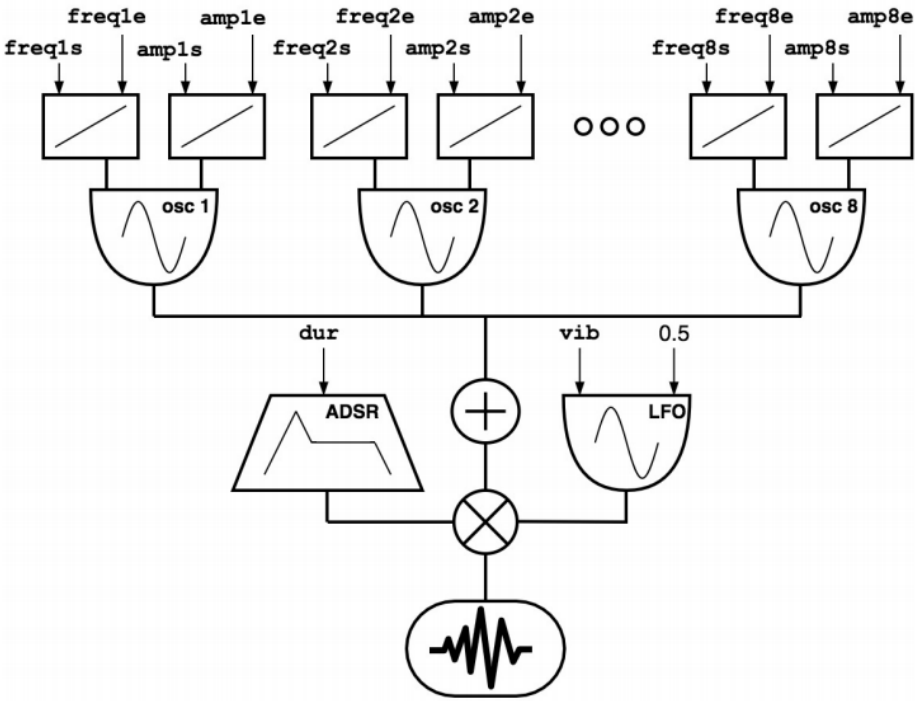


Fig. 4. qSyn's layout.

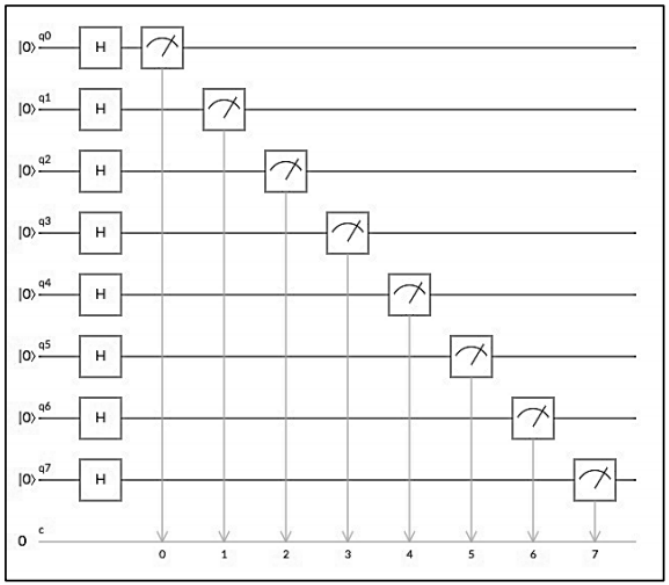


Fig. 5. The quantum hyper-die circuit. The operator, or gate H (for Hadamard) puts all 9 qubits in superposition.

qSyn (Quantum Synthesizer)

Implementation (cont.)

Figure: Retrieval of Oscillator Frequences

Triplet Code	Binary	Decimal	Parameter	Retrieved Value
(c_8 c_7 c_6)	000	0	`freq1s`	55.0 Hz
(c_6 c_7 c_8)	000	0	`freq1e`	55.0 Hz
(c_5 c_4 c_3)	001	1	`freq2s`	369.99 Hz
(c_3 c_4 c_5)	100	4	`freq2e`	466.16 Hz
(c_2 c_1 c_0)	001	1	`freq3s`	349.23 Hz
(c_0 c_1 c_2)	100	4	`freq3e`	87.3 Hz
(c_7 c_6 c_5)	000	0	`freq4s`	92.49 Hz
(c_5 c_6 c_7)	000	0	`freq4e`	92.49 Hz
(c_4 c_3 c_2)	000	0	`freq5s`	435.53 Hz
(c_2 c_3 c_4)	000	0	`freq5e`	435.53 Hz
(c_8 c_5 c_2)	000	0	`freq6s`	440.0 Hz
(c_2 c_5 c_8)	000	0	`freq6e`	440.0 Hz
(c_7 c_4 c_3)	011	3	`freq7s`	2200.0 Hz
(c_3 c_4 c_7)	110	6	`freq7e`	1385.91 Hz
(c_6 c_3 c_0)	001	1	`freq8s`	2354.63 Hz
(c_0 c_3 c_6)	100	4	`freq8e`	3960.0 Hz

Figure: Retrieval of Oscillator Amplitudes

(d_8 d_7 d_6)	001	1	`amp1s`	0.08
(d_6 d_7 d_8)	100	4	`amp1e`	0.14
(d_5 d_4 d_3)	011	3	`amp2s`	0.12
(d_3 d_4 d_5)	110	6	`amp2e`	0.18
(d_2 d_1 d_0)	000	0	`amp3s`	0.06
(d_0 d_1 d_2)	000	0	`amp3e`	0.06
(d_7 d_6 d_5)	010	2	`amp4s`	0.1
(d_5 d_6 d_7)	010	2	`amp4e`	0.1
(d_4 d_3 d_2)	000	0	`amp5s`	0.06
(d_2 d_3 d_4)	000	0	`amp5e`	0.06
(d_8 d_5 d_2)	010	2	`amp6s`	0.1
(d_2 d_5 d_8)	010	2	`amp6e`	0.1
(d_7 d_4 d_3)	110	6	`amp7s`	0.18

qSeq (Adaptive Musical Sequencer)

Brief Overview

- The qSeq takes as input some melody, processes it, and outputs a sequence of notes based on ...

Extracted Features

1. Pitches
2. Duration
3. Loudness

- Transition matrices constructed based on extracted features.

- These matrices will form the basis of the quantum circuit that implements qSeq.

qSeq (Adaptive Musical Sequencer)

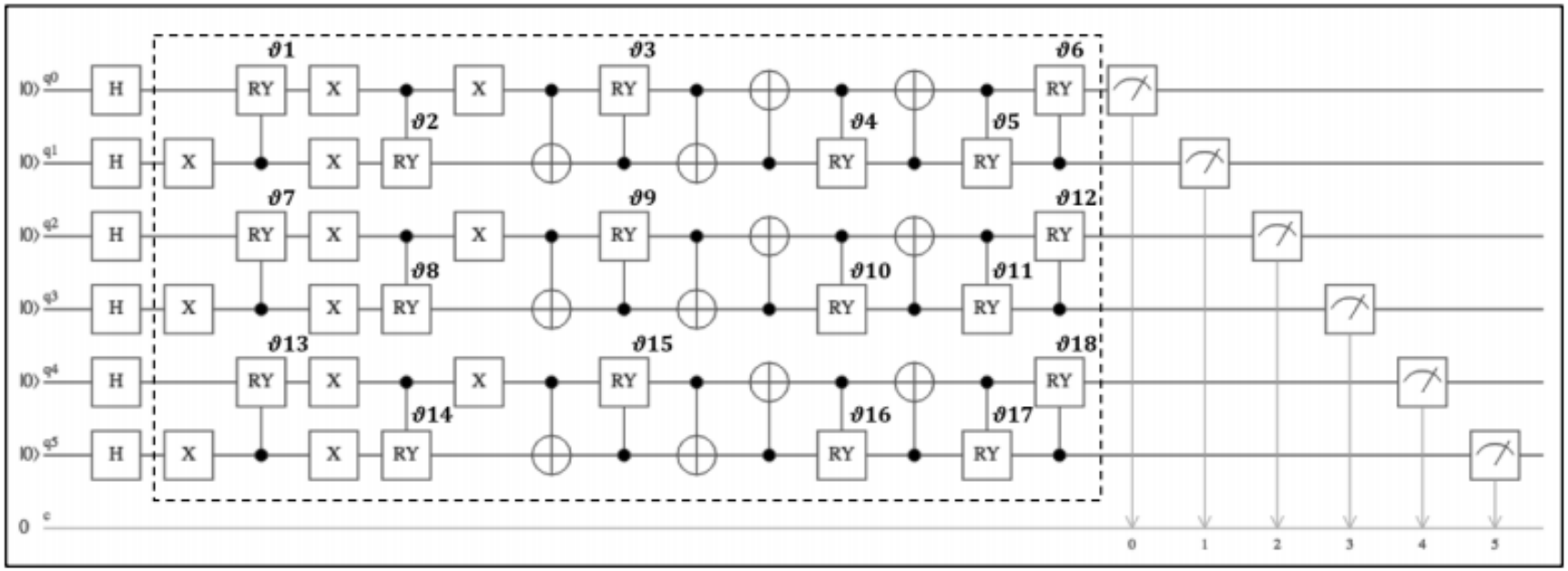
Implentation

- Can't implement transition matrices as is for quantum state evolution
  - Why? $\rightarrow$ They are not unitary...
- One approach is to convert our transition matrices (doubly-stochastic matrices) into unistochastic matrices.
- In this paper, transition matrices are converted into a list of angles to be implemented by the $R_y(\theta)$ quantum gate.
  - $[v_1, v_2, ..., v_{18}]$

qSeq (Adaptive Musical Sequencer)

qSeq Circuit

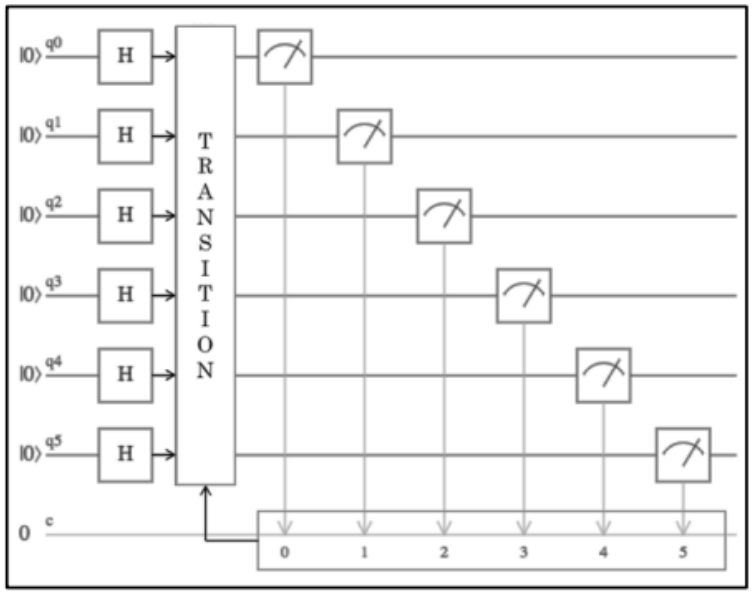
Figure: qSeq quantum circuit



qSeq (Adaptive Musical Sequencer)

Implementation

Figure: Simplified view of qSeq quantum circuit



Circuit Implementation Steps

1. Apply Hadamard gate to all qubits
2. Apply "transition" gate to qubits
3. Perform Measurements on all qubits
4. Feed measured metrics back into transition gate for new note.

qSeq (Adaptive Musical Sequencer)

Finer Details

- Input Melody

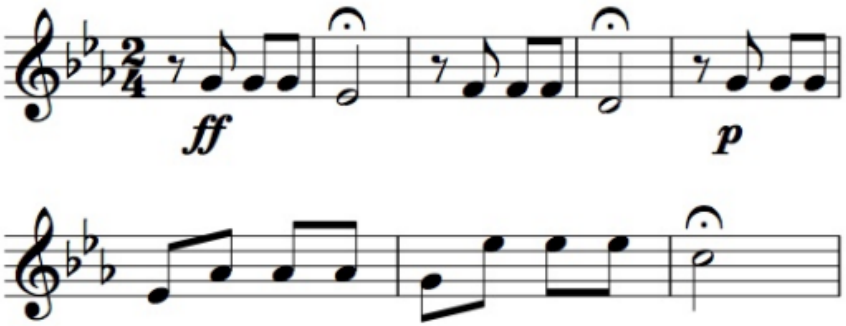


Fig. 10. The opening of Beethoven's 5th symphony.

- Extracted Features

P = [67, 67, 67, 63, 65, 65, 65, 62, 67, 67, 67, 63, 68, 68, 68, 67, 75, 75, 75, 72]

D = [298, 301, 302, 1798, 302, 297, 301, 1799, 302, 303, 296, 302, 297, 302, 298, 301, 297, 301, 297, 1799]

L = [113, 113, 113, 105, 113, 113, 113, 107, 61, 61, 61, 57, 64, 63, 63, 61, 70, 68, 67, 60]

- Extracted Features (Altered)

P' = [67, 67, 67, 63, 65, 65, 65, 62, 67, 67, 67, 63, 67]

D' = [298, 301, 302, 1798, 302, 301, 302, 302, 302, 298,

L' = [113, 113, 113, 105, 113, 113, 113, 107, 61, 61, 61,

- Doubly-Stochastic Matrix

P'	67	63	65	62
67	0.131	0.858	0.001	0.009
63	0.192	0.03	0.724	0.053
65	0.001	0.005	0.244	0.750
62	0.676	0.106	0.031	0.188

Fig. 12. Bistochastic matrix for note pitches.

qSeq (Adaptive Musical Sequencer)

Finer Details

- Recall our list of angles...
  - $[v_1, v_2, ..., v_{18}]$
  - Why are there exactly 18 rotation angles listed? $\rightarrow$ 3 metrics (pitch, duration, loudness) $\times$ 6 degrees of freedom $=$ 18 rotation angles.
- Rotation Angles (based on above example)
  - $\Theta P' = [243, 197, 243, 186, 180, 249]$
  - $\Theta D' = [237, 203, 128, 203, 169, 249]$
  - $\Theta L' = [220, 180, 157, 180, 140, 123]$
- $\therefore \Theta = [243, 197, 243, 186, 180, 249, 237, 203, 128, 203, 169, 249, 220, 180, 157, 180, 140, 123]$

qSeq (Adaptive Musical Sequencer)

Simulating the Process

- **Round 1:**
  Initial default quantum state:
  $H(|0\rangle) \otimes H(|0\rangle) \otimes H(|0\rangle) \otimes H(|0\rangle) \otimes H(|0\rangle) \otimes H(|0\rangle$
  Measurement Results: $M = [0, 1, 1, 0, 1, 0]$
- **Round 2:**
  Quantum State: $|0\rangle|1\rangle|1\rangle|0\rangle|1\rangle|0\rangle$
  Measurement Results: $M = [1, 1, 1, 1, 0, 0]$
- **Round 3:**
  Quantum State: $|1\rangle|1\rangle|1\rangle|1\rangle|0\rangle|0\rangle$
  Measurement Results: $M = [0, 0, 0, 0, 0, 0]$
- **Round 4:**
  Quantum State: $|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle|0\rangle$
  Measurement Results: $M = [1, 0, 1, 0, 1, 0]$

qSeq (Adaptive Musical Sequencer)

Choosing Note Characteristics

- Upon measuring quantum state, we have an array of the following form:

$$M = [m_0, m_1, m_2, m_3, m_4, m_5]$$

- Pairs of elements define binary codes related to characteristics of generated note :
  - (m_0, m_1) - Establish which pitch set note will be retrieved from
  - (m_2, m_3) - Establish note duration.
  - (m_4, m_5) - Establish note pitch

The Quantum Composer

Putting it all together - qSyn + qSeq

Figure: Paper's implementation of their *Zeno* Composer