

"The Fourth Man," for Fretless Electric Guitar and Real-Time Electronics

Adam James Wilson*

New York City College of Technology, City University of New York
awilson@citytech.cuny.edu

Abstract

"The Fourth Man" is the latest in a series of pieces for electric guitar and "automatic improvisation" software. Prior works in the series have been performed in Europe, East Asia, and throughout the United States. While the guitarist performs, pitched accompaniment is developed from real-time analysis of the player's output. In earlier works, a simple probability model was used to traverse a factor oracle model of the player's live performance to produce idiomatic recombinations of material. In "The Fourth Man," this process is augmented with a novel pattern discovery algorithm leveraging lossy compression to exploit repeated motives in the soloist's output.

1 SYNOPSIS

"The Fourth Man" involves equal parts live improvisation, offline algorithmic composition, and real-time generative processes.

1.1 Pre-composition

Percussion parts are composed using the author's DrumGen software. This software accepts a kernel of composed percussion music in a prescribed JSON format. Each kernel is composed of multiple linear streams that appear simultaneously. DrumGen provides functions to transform these streams by rearranging segments of individual streams in time, superimposing different streams to form new simultaneities, performing rhythmic augmentation or diminution, building sections that prioritize events with certain criteria — such as instrumentation or loudness, rearranging the playback order of musical events inside streams, exchanging instrumentation between streams, etc. Not only is the kernel upon which DrumGen acts pre-written, but the sections of music that are generated by DrumGen are pre-generated, run through multiple cycles of audition, re-parameterization, and realization. DrumGen outputs multichannel MIDI files which are fed to various software instruments in real-time. The percussion parts provide a framework for the overarching structure of the piece, and sequences of events generated in real-time are tied to onsets in the percussion parts.

1.2 Generative processes

Two generative processes are at play. One is tied to a factor oracle model of the performer's live input, and the other leverages a catalog of maximal repeated patterns below a threshold of length N .

1.2.1 Factor oracle

The factor oracle is a directed acyclic word graph capable of expressing at least all of the substrings of a given word. The oracle can be built incrementally, in $O(n)$ time and space. A factor oracle, as we're using it, has musical events as edge labels connecting the states of the graph. The first event labels the forward transition between states

zero and one, and each following event labels a forward transition between consecutive states. States may also have forward transitions to one or more non-consecutive states. Each transition of the latter kind ends a sequence of transitions that shares a suffix with a substring of the original word terminated by the state to which the transition points (Wilson, 2016).

We exploit these skip-state transitions to recombine musical materials in novel ways. As the performer plays, a factor oracle model of the player's input is built in real-time. In this case, each input element is a single integer representing a cross-alphabet of pitch-class and rhythm. At the same time the model is constructed, a parallel process walks the model, selecting forward transitions that may jump via common suffix from one region of the graph to another. This process emits a stylistically relevant yet novel sequence of musical events for accompaniment. To circumvent the acyclic nature of the factor oracle, we harness backward-pointing edges called suffix links, a side-effect of the factor oracle construction process, to turn the factor oracle into a cyclic graph for continuous output. We can also reverse segments of the input by making forward transitions into bi-directional edges. The level of coherence between the output and input is controlled by a probability value that weights forward link traversal (more coherent) against suffix link traversal (less coherent).

The factor oracle implementation used here is an open-source C external for Cycling '74 Max and Pure Data written by the author. It works with auxiliary objects that encode and decode cross-alphabet values.

1.2.2 Lossy compression for motive discovery

A second Max external, MD (for 'motive discovery'), also developed by the author, is used to count and return the maximal musical motives appearing in the performer's output. This process also has analytical and generative components. On the analysis side, we limit ourselves to looking at segments of no longer than 16 musical events. An event is considered a pitch plus an interval of time that follows until the next event (an inter-onset interval).

Since our input is assumed to be free improvisation, we can't count on a set of scores or recordings of related material to inform our analysis. So we compress the data

*<https://adamjameswilson.info>

by stripping it down to the most perceptually meaningful features; the more we compress, the more likely we are to find repeated material. In this case, we privilege contour over precise pitch and precise locations in time.

MD treats each potential motive as a pair of complete graphs — one for pitch, one for inter-onset interval — whose edges are the deltas between all pairwise combinations of N consecutive note events, where $3 \leq N \leq 16$. N is an upper bound: matches are reported for prefix sub-graphs of order 3 through N , so a shorter shared opening still registers as a motive. A graph is labelled a motive if it occurs at least twice and is maximal. Each delta is compressed to one of three states — increase, decrease, or no change — stored as a two-bit crumb. The user can query MD at any moment for the top motives by size or occurrence count. Graphs are returned as hexadecimal strings and must be expanded by an auxiliary Max object (under development).

Figure 1 shows an abstraction of the data structure used in MD. We compare these structures in an online LZ77-style pattern discovery algorithm modified to catalog all of the maximal motives in the performance (Ziv and Lempel, 1977; Gailly and Adler, 2026). The collected motives are decompressed in real-time and transformed according to some compositional heuristics programmed in Max. These are fed to virtual instruments to provide further accompaniment. For more details about MD, see Wilson (2026).

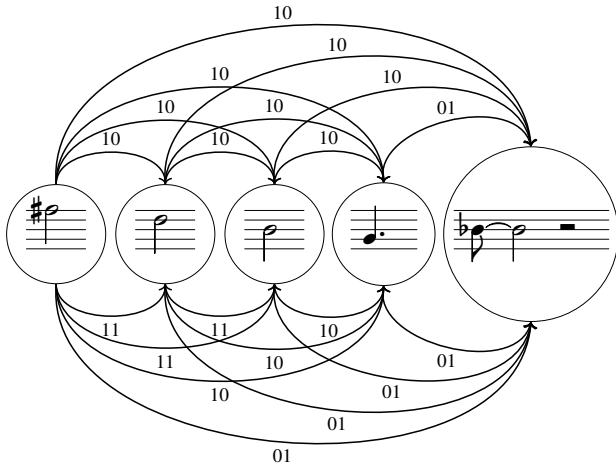


Figure 1: Crumb assignments for pitch deltas (top edges) and inter-onset interval (IOI) deltas (bottom edges). Music is taken from the beginning of the theme in John Coltrane’s “Giant Steps” (Coltrane, 1959).

REFERENCES

- Coltrane, J. (1959). Giant Steps. Atlantic Records (SD 1311). Recorded May 4 & 5 and December 2, 1959.
- Gailly, J.-I. and Adler, M. (2026). zlib: A massively spiffy yet delicately unobtrusive compression library. <https://zlib.net>. Reference implementation; hash-chain longest-match logic in `deflate.c`.
- Wilson, A. J. (2016). factorOracle: An extensible max external for investigating applications of the factor oracle automaton in real-time music improvisation. In Pasquier, P., Bown, O., and Eigenfeldt, A., editors, *Proceedings of the 4th International Workshop on Musical Metacreation (MUME 2016)*. Held at the Seventh International Conference on Computational Creativity, ICC3 2016.
- Wilson, A. J. (2026). Fast discovery of motivic patterns in symbolic music via lossy compression. In *Proceedings of the 7th Conference on AI Music Creativity (AIMC 2026)*, Berlin, Germany. September 16th–18th.
- Ziv, J. and Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343.