CHARLIE MARKOV: AN ALGORITHMIC APPROACH TO THE STYLE OF CHARLIE PARKER

Jorge Elías Variego, PhD

Valley City State University

www.jorgevariego.com

Abstract

The present work is based on the hypothesis that the improvisational style of saxophonist Charlie Parker is condensed in the first chorus of his solo in his tune Confirmation.[1] In order to verify this initial question, the study proposes a series of probabilistic analysis from the solo that are integrated in an algorithm in SuperCollider as a Markov chain of the first order.

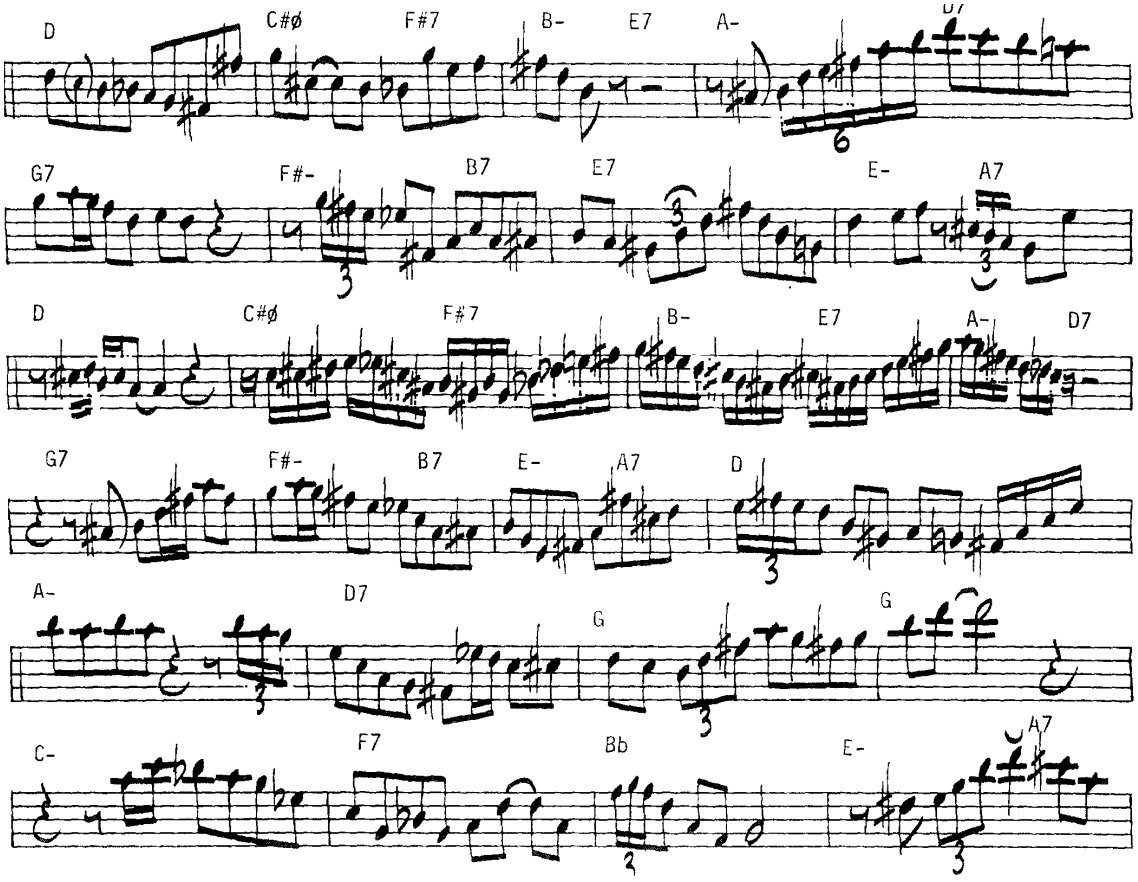

Figure 1. Excerpt from Parker's first chorus in his tune *Confirmation*. The excerpt is transposed to written pitches in Eb. The figure corresponds to the audio file 1.

[1] According to the transcription in the *Omnibook.*

Markov chains and studies of style

Markov chains of the first order are probabilistic models that are used to predict the evolution and behavior in the short and long term for certain systems (in this case the system or *corpus* [2] *is* Parker's solo). They are characterized by being "memoryless", as the changes of state depend exclusively on the current state and not on the sequence of events that precede it. In the succession of pitches, the "following note" (change of state) depends exclusively on the "current note" (current state) and not on any of the previous ones. This statement reconfirms the advantages of Markov's method for the study of musical style, especially when it is implemented for *monodic* constructions.[3] Markov Models, like the generative grammars, are replacement systems, which structures only allow the description of contextual dependencies through the transition probabilities of symbols in direct succession (i.e. a series of "notes" in a melodic construction in the solo). This structural feature of Markov chains do not correspond exactly with the complete musical information as it has a layered vertical dimension (i.e. relations between the voices of a polyphonic texture). (Nierhaus, 2010, 81) [4]

Due to the essence of the *markovian* method, the type and quality of the result yielded by the algorithm depends largely on the *corpus'* qualities [...] (Nierhaus, 2010, 81). Following a logical reasoning after this premise, we could say that the more comprehensive the corpus, the more *stylistic* the result offered by the algorithm. However, the aim of this study is not to simply reaffirm that concept but to introduce a new question to it, sparked by the fact that in the particular case of the improvisational style of Charlie Parker, the character of the corpus – more or less complete – seems not to have a significant influence on the results yielded by the algorithm.

Algorithmic application of the probabilistic data

The probabilities of transition between states are integrated into the algorithm autonomously. State changes on rhythmic figures are considered regardless of the pitches and vice-versa. This methodology allows us to maintain control over every parameter independently, coding its

[2] The term *corpus* is used in a manner analogous to luguistics and refers to the set of turns, phrases, motifs, and other musical devices that can be considered representative of the musical style of an artist.

[3] Note that in the initial application of the probabilistic method, Markov conducted a study on the succession of vowels and consonants in a group of 20,000 letters in the poem *"Eugeny Onegin"* by A. S. Pushkin.

[4] That is why this study only addresses issues of melodic construction.

synchronization with routines that use global variables. [5]

The classic example of a *markovian* chain is "the drunken walk", in which the next state is +1 or -1 from the current state. In the example below, the "drunken walk" is applied exclusively to the succession of pitches (-1 = descending semitone, +1 = ascending semitone). The rhythmic figures are already determined by the percentages taken from the solo (data from Table 1) but do not follow a *markovian* model. [6] Consider the code:

```
(// “Drunken walk” with the probabilities of the rhythmic figures extracted from the solo

SynthDef("sine",{arg frecuencia, canal = 0 , duration = 0.1, amp = 1;

        var env, out, envsig ;
        env  = Env.new( [ 0.0, 1.0, 1.0, 0.0 ], [ 0.101, 0.6, 0.299 ],'sine');
        envsig = EnvGen.ar( env, timeScale: duration, levelScale: 0.1, doneAction: 2  ) ;
        out = { Pulse.ar(frecuencia, Line.kr(0.01,0.5, duration / 4) , envsig * amp) };
        Out.ar(canal, [out, out] )  ; // multichannel expansion
}).send(s) ;
r = Routine.new({
                ~nota = 60;
                ~tempo = 0.3; // multiplier that lets us control the “tempo”
                inf.do({

                        ~ritmo = ([0.5, 1, 0.25, 0.33, 0.17, 1.5, 2.5].wchoose([101, 4, 56, 9, 15, 1, 1
].normalizeSum)); // is it enough to apply the data from the tables in this way?
                        ~nota = ~nota + ([1, -1].choose); // <- "drunken walk"
                        ~nota.postln;
                        ~amplitude = 1.0;
                        ~ritmo.postln;
```

[5] This algorithmic design strategy is useful to control tempo. Additionally, the functional structure of the algorithm will also let us dispose freely and independently (rhythms and pitches) of the probabilistic data extracted from the solo.

[6] This example corresponds to audio file 2.

```
                    Synth.new("sine", [\frecuencia, ~nota.midicps, \duration, (~ritmo * ~tempo),
\amp, ~amplitude]);
                    (~ritmo * ~tempo).wait;
                });
});
a = Routine.new({
        r.play;

}).play; // This routine only plays the routine (r) but it will – later – let us control independently
the routines that handle pitches and rhythmic figures (nesting routines).
```

While the rhythmic data extracted from the transcription is correct, its algorithmic application *sounds* unsatisfactory because the rates of occurrence of the rhythmic figures do not take into account the "current state" (i.e. the current rhythmic figure) to determine the likelihood of the next figure, but merely return a rhythm according to the corresponding percentages. If we only apply the percentages of occurrence of events, we are respecting their percentages of occurrence (in this case the rhythmic figures), disregarding the probabilities in their succession, and thus missing a key element of Parker's style. We know that the there is only a 54% of eighth notes (101 out of a total of 187 figures), which is incorporated in the code. However, because the code does not consider transitions, if the current state is an eighth note, we do not know what are the chances that the figure that follows is a quarter, a sixteenth, an eighth, and so on. From a different perspective, we know that the chances remain immobile regardless of the current state.

Table 1. Amount of rhythmic figures extracted from the solo.

Proportional duration (♩ = 1)	**Quantity**
0.5 = ♪	101
1 = ♩	4
0.25 = 𝅘𝅥𝅯	56

0.33 = ♪triplet	9
0.17 = 𝅘𝅥𝅯sextuplet	15
1.5 = 𝅗𝅥.	1
2.5 = 𝅘𝅥𝅯𝅗𝅥	1
Total	187

This principle is also applicable to the realm of pitches as we can appreciate in the following table. Likewise, this chart simply offers quantitative data in the realm of intervals.

At first glance, we can make several preliminary reflections on the intervallic content of the solo: 1) the melodic contour is not very angular, that comes from the primacy of the intervals of less than a fourth, 2) stepwise motion prevails, either by semitone or whole tone in both directions, 3) the number of minor thirds in both directions (arpeggios of diminished chords or dominant 7th structures?) is another significant feature, 4) the amount of major thirds is smaller than the minor thirds, reaffirming the primacy of diminished arpeggios or major chords with minor sevenths (dominant structures). These preliminary considerations become vital at the moment of the aural screening of the algorithm.

Table 2. Amount of intervals extracted from the solo, ordered according to their size (in semitones).

Charlie Parker's solo (1st chorus) in *Confirmation*	
Interval (semitones)	**Quantity**
+1	24
-1	25
+2	22
- 2	36
+ 3	21
- 3	20
+ 4	7
- 4	11
+ 5	2
- 5	5
+ 6	0
- 6	1
+ 7	1
- 7	1

+ 8	2
- 8	0
+ 9	4
- 9	1
+ 10	0
- 10	0
+ 11	0
- 11	0
+ 12	1
Total	183

Now let's see the excerpt of code that incorporates the data from both tables as simple percentages of events.[7]

```
inf.do({
    ~nota = ([-1,1,-2,2,-3,3,-4,4,-5,5,-6,6,-7,7,-8,8,-9,9,-10,10,-11,11,-12,12].wchoose([25,24,36,22,20,21,11,7,5,2,1,0,0,1,0,2,1,4,0,0,0,0,0,1].normalizeSum)); // instead of the "drunken walk" from the previous example
    ~nota.postln;
```

In the infinite .do loop, the value of the global variable ~note is assigned (in each iteration of the loop) at the rates taken from the table in the form of a normalized sum. While it is likely that the aural result will reflect some of the properties of the studied style, it is also possible that the program will not necessarily return *parkerian* melodic constructions; in other words, the algorithm could return sequences of pitches that respect the percentages of probability, but not the style.

To obtain a more *stylistic* result we would have to consider the transitions in both the realms of pitch and rhythm. These transitions will give us the data needed to create a *markovian* chain. The transition probabilities of a *markovian* chain are constant over time, the data tables reflect the probabilities of succession of events taken from Parker's solo, or what is the degree of probability that a pitch *x* or rhythmic figure *y* (i.e. next state) will follow the current state (always taking into account the data from the solo).

[7] This example corresponds to audio file 3 (it uses the same SynthDef as audio file 2).

Table 3. Succession of rhythmic figures.

Total ♪ = 101		**Total 𝅘𝅥𝅯 = 56**		**Total ♪ triplet = 12**	
♪ -> ♩	3	𝅘𝅥𝅯 -> 𝅘𝅥𝅯	49	♪ triplet -> ♪	3
♪ -> 𝅘𝅥𝅯 sextuplet	2	𝅘𝅥𝅯 -> ♪	5	♪ triplet -> ♩	1
♪ -> ♪ triplet	3	𝅘𝅥𝅯 -> ♩.	1	♪ triplet -> ♪ triplet	8
♪ -> 𝅘𝅥𝅯	4	𝅘𝅥𝅯 -> rest	1		
♪ -> ♪	84				
♪ -> rest	5				

Total 𝅘𝅥𝅯 sextuplet = 15		**Total ♩ =**	
𝅘𝅥𝅯 sextuplet -> ♪	4	♩ -> ♪	all ♩ followed by ♪
𝅘𝅥𝅯 sextuplet -> 𝅘𝅥𝅯 sextuplet	11	**Total 𝅗𝅥 + ♪ =**	
Total ♩. =		Followed by rest 𝅘𝅥𝅯 or 𝅗𝅥 + ♪	
Followed by rest ♪ or 𝅘𝅥𝅯 in equal %			

The following code example shows one way to use conditional structures to evaluate the current rhythmic state and – according to it – decide which one will follow. It must be stated that there are other solutions, as the Markov chains can be represented with a graphic of transitions or with a matrix (Nierhaus, 2010, 68).

```
if (~ritmo == 0.5, {~ritmo = ([1, 0.17, 0.33, 0.25, 0.5, 2.5].wchoose([3, 2, 3, 4, 84,
```

```
2].normalizeSum)); ~amplitude = rrand(0.9, 1.0)},
            // if it is not an eight
{if (~ritmo == 0.25, {~ritmo = ([1.5, 0.5, 0.25].wchoose([1, 5, 49].normalizeSum))},
            // if it is not a 16th
{if (~ritmo == 0.33, { ~ritmo = ([0.5, 1, 0.33].wchoose([3, 1, 8].normalizeSum))},
            // if it is not a triplet
{if (~ritmo == 0.17, {~ritmo = ([0.5,  0.17].wchoose([25, 75].normalizeSum))},
            // if it is not triplet of 16ths
{ if (~ritmo == 1, {~ritmo = 0.5},
            // if it is not a quarter
{if (~ritmo == 1.5, {~ritmo = [0.25, 0.5].choose; ~amplitude = 0},
            // if it is not a dotted quarter
{if (~ritmo == 2.5, {~ritmo = [0.25, 1.5].choose; ~amplitude = 0 }
 )};
                                    )})})}
                                                    )}; )});
```

Table 4. Succession of intervals (in semitones).

Succession of +1 (total +1 = 24)		**Succession of -1 (total -1 = 25)**		**Succession of +2 (total +2 = 22)**		**Succession of -2 (total -2 = 36)**	
+1	4	+1	2	+1	4	+1	2
-1	2	-1	3	-1	0	-1	3
+2	7	+2	0	+2	4	+2	0
-2	2	-2	9	-2	7	-2	9
+3	3	+3	3	+3	3	+3	3
-3	2	-3	1	-3	2	-3	1
+4	1	+4	0	+4	0	+4	0
-4	2	-4	1			-4	1
+5	0	+9	2			+9	2
-5	0	-9	1			-9	1
+6	0	+12	1			+12	1
-6	1						

Succession of +3 (total +3 = 21)		Succession of -3 (total -3 = 20)		Succession of +4 (total +4 = 7)		Succession of -4 (total -4 = 11)	
+1	0	+1	6	+3	3	+1	2
-1	1	-1	1	-4	1	-1	3
+2	3	+2	5	+7	1	+2	0
-2	3	-2	1			-2	9
+3	5	+3	3			+3	3
-3	5	-3	0			-3	1
+4	5	+4	0			+4	0
-4	0	-4	2			-4	1
+9	1	-5	2			+9	2
						--9	1
						+12	1

Succession of +5 (total +5 = 2)		Succession of -5 (total -5 = 4)		Succession of -6 (total -6 = 1)		Succession of +7 (total +7 = 2)	
-1	1	+1	1	-2	1	+2	1
+3	1	+3	1			-2	1
		-4	1				
		+9	1				

Succession of +8 (total +8= 2)		Succession of +9 (total +9 = 3)		Succession de -9 (total -9 = 1)		Succession of +12 (total +12 = 1)	
+1	1	-1	1	+3	1	+1	1
+2	1	-3	1				
		-5	1				

The code regarding the evaluation of the pitches and intervals is similar:

```
        // ascending semitone
if (~nota == 1, {~nota = ([1,-1,2,-2,3,-3,4,-4,5,-5,6,-
6].wchoose([4,2,7,2,3,2,1,2,0,0,0,1].normalizeSum))},
        // descending semitone
```

```
{if (~nota == -1, {~nota = ([1,-1,2,-2,3,-3,4,-4,9,
9,12].wchoose([2,3,0,9,3,1,0,1,2,1,1].normalizeSum))},
        // ascending major second
{if (~nota == 2, {~nota = ([1,-1,2,-2,3, 3,4].wchoose([4,0,4,7,3,2,0].normalizeSum))},
        // descending major second
{if (~nota == -2, {~nota = ([1,-1,2,-2,3,-3,4,-4,9,-
9,12].wchoose([2,3,0,9,3,1,0,1,2,1,1].normalizeSum))},
        // ascending minor third
{if (~nota == 3, {~nota = ([1,-1,2,-2,3,-3,4,-4,9].wchoose([0,1,3,3,5,5,5,0,1].normalizeSum))},
        // descending minor third
{if (~nota == -3, {~nota = ([1,-1,2,-2,3,-3,4,-4,5].wchoose([6,1,5,1,3,0,0,2,2].normalizeSum))},
        // ascending major third
{if (~nota == 4, {~nota = ([3,-4,7].wchoose([3,1,1].normalizeSum))},
        // descending major third
{if (~nota == -4, {~nota = ([1,-1,2,-2,3,-3,4,-4,9,
9,12].wchoose([2,3,0,9,2,1,0,1,2,1,1].normalizeSum))},
        // ascending fourth
{if (~nota == 5, {~nota = ([-1, 3].wchoose([1,1].normalizeSum))},
        // descending fourth
{if (~nota == -5, {~nota = ([1, 3, 9, -4 ].wchoose([1,1,1,1].normalizeSum))},
        // ascending tritone
{if (~nota == -6, {~nota = -2 },
        // ascending fifth
{if (~nota == 7, {~nota = [2,  -2].choose },
        // ascending minor sinth
{if (~nota == 8, {~nota = [2, 1].choose},
        // ascending major sixth
{if (~nota == 9, {~nota = [-3, -5, -1].choose},
        // descending major
{if (~nota == -9, {~nota = 3},
        // ascending octave
```

```
{if (~nota == 12, {~nota = 1},
                                    )} )} )} )} )} )} )} )} )} )} )} )} )} )} )} )});
```

The eight note with *swing* is a characteristic element of jazz that we can embed in our algorithm. The following is one of the many possible solutions to the problem. This section of the code goes at the end of the loop. [8]

```
swingDuracion = Pseq([0.53, 0.47], inf).asStream; // The variable as a pattern of durations that evoke the feel of a light swing. Values such as [0.6, 0.4] would return a heavier swing.
(
        Here goes the rest of the code …
                                    )
~notaDespues.postln;
            sampleFileToPlayNow = ~notaDespues.asSymbol;

if (~ritmo == 0.5, {~ritmo = (swingDuracion.next + rand2(0.02))}); // <- If it is an eight note, the duration is according to the value of swingDuracion. The random number generator rand2(0.02) adds or subtracts a small random number that is intended to recreate the imperfections of the human touch.
            ~ritmo.postln;

            Synth.new("piano", [\out, 0, \bufnum, buffer.at(sampleFileToPlayNow).bufnum, \duration, (~ritmo * ~tempo), \envTimeScale, 3, \pan, 0, \db, ~amplitude * rrand(1.0, 1.1)]); // <- The amplitude also has a small random multiplier in order to "humanize" the touch of the algorithm.
            (~ritmo * ~tempo).wait; // The rhythmic figures according to the global variable ~tempo.
if ((~ritmo < 0.62) && (~ritmo > 0.5), {~ritmo =0.5},
            {if ((~ritmo < 0.50) && (~ritmo > 0.35), {~ritmo = 0.5}); // <- These two
```

[8] This example corresponds to the audio file 4 (which uses a different SynthDef, a sampler based on piano sounds, not included in this study).

conditionals evaluate the value of ~ritmo and reassign it to 0.5 before it is reevaluated through the chain of conditionals of the .do loop.

});

Conclusion

This paper presents a new interrogation to the study of style and algorithmic composition, confirming the effectiveness of Markov chains in the subject, but questioning the relationship between *corpus* and output for the study of the style of certain artists. If the aural result returned by the algorithm is totally or partially accepted as representative the *parkerian* style, we could then formulate new questions: Is Parker's improvisational style contained in any of his solos? Are there substantial probabilistic changes in Parker's solos from different moments of his career? Are the *markovian* chains the most appropriate analytical method for this type of study? Could we create generative grammars based on the same *corpus* and obtain similar results?

"Bach developed a kind grammar that I merely picked up, as could anyone who wished to. And then, armed with this grammar, I – just like anyone with a bit of musical talent – can easily compose any number of pieces in perfect Bach style. It takes no genius, believe me it's all straightforward stuff. The only place where genius was involved was coming up with the grammar." [9]

[9] According to a conversation between Cope and Rowell (Cope, 2001, p. 57).

References

1. Wilson S., Cottle D., Collins N. (2011) *The SuperCollider Book.* Massachusetts Institute of Technology, Boston, US.
2. Cope D., (2005) *Computer Models of Musical Creativity.* Massachusetts Institute of Technology, Boston, US.
3. Cope D., (2001) *Virtual Music: Computer Synthesis of Musical Style.* Massachusetts Institute of Technology, Boston, US.
4. Nierhaus G. (2010) *Algorithmic Composition: Paradigms of Automated Music Generation.* Springer-Verlag/Wien, Germany.
5. Russell J., Cohn Ronald (2012) *Algorithmic Composition.* Lennex Corp., Scotland, UK.