COMPUTER COUNTERPOINT

Kemal Ebcioğlu
Department of Computer Science
SUNY at Buffalo
4226 Ridge Lea Road
Amherst, NY 14226

This paper is about a program that writes two part florid counterpoint exercises. The goal in two part florid counterpoint, a species of strict counterpoint, is to write a melody in rhythmically free style (called a Contrapunctus) that matches the given Cantus Firmus, and complies with all the applicable rules of strict counterpoint. Our program's style is that of a typical "conservatory student"; It can roughly be described as a mixture of the styles advocated by Charles Koechlin[1] and Joseph Marx[2]. The program, written in Lisp, is given a Cantus Firmus and a set of rules as input. It then outputs as many solutions as it can within a given time limit.

In view of the complexity of rules of strict counterpoint even for this modest exercise, we had to develop a method that would enable us to program the counterpoint rules in a conceptually simple and modular manner. We shall explain this method in the forthcoming paragraphs.

We represent the Contrapunctus (let us call it the melody) as a Lisp list. Each rule is represented as a very restricted Lisp function, which, when viewed as a program, scans the list representing the melody from left to right, and halts saying either "No(Yes), there is a 'mistake' in the melody", or "Yes(No), the melody is OK with respect to this rule". Some recursive theoretic properties of these functions, called normal characteristic functions, have been investigated in [3]. An informal definition of normal characteristic functions will be given in the Appendix.

The usefulness of normal characteristic functions (NCF's) lies in the following facts:

1) We can easily write an NCF for each simple property of a melody. For example, we can write an NCF that says "yes" if and only if the melody begins with a rest. Similarly, we can write another NCF that says "yes" if and only if the range of the melody exceeds an eleventh. Indeed, for the sake of an example let us suppose we have written NCF's for each of the following properties: (we will see what we can do with them below)

Begin_with_rest (the melody begins with a rest)

Bad_range (the range of the melody exceeds an eleventh)

Exposed_octaves (the given C.F. and the melody reach an octave with direct motion)

2) Normal characteristic functions turn out to be a Boolean algebra [3]. Hence from the NCF's for the simple properties of melodies, we can construct another NCF that corresponds to an arbitrary combination of these under 'and's, 'or's and 'not's. E.g. from the three simple NCF's given above we can construct an NCF for the following "composite" property:

Begin_with_rest and not(Bad_range or Exposed_octaves) (1)

This "constructed" NCF will say "yes" if and only if the melody has the "composite" property given in (1), which is possible if and only if the melody begins with a rest, and its range is OK, and it does not produce exposed octaves with the Cantus Firmus. We do not do any reprogramming for obtaining this NCF, we merely put in the 'and's and 'not's and 'or's and an algorithm does the rest [3].

3) There is an enumeration algorithm that will output one by one (theoretically all) the melodies having the property corresponding to a given NCF. This NCF can of course correspond to a "composite" property, as in (1). Hence we can feed the NCF we constructed for (1) to this enumeration algorithm and let it run for a reasonable amount of time (2 minutes on a 370/145 in our case). We will then get a lot of computer generated melodies that comply exactly with our specification, as given in sentence (1). The outputs will all begin with a rest, their ranges will be within an eleventh, and they will not produce any exposed octaves with the Cantus Firmus (It is certainly doubtful whether they will sound any good, but after all, strict counterpoint does not consist only of these three simple rules!). The enumeration algorithm uses a search strategy that influences the order of enumeration of the melodies. This search strategy is easily modifiable by the user, so that he/she has a chance to have the "preferred" melodies enumerated before time runs out.

In the implementation of our program, the same sequence of actions was followed: we wrote a simple NCF for each property appearing in the rules of counterpoint. We then

constructed the necessary Boolean combination of these NCF's and then we fed the result to the enumeration algorithm. Since working with individual NCF's simplified matters immensely, we were able to program the "counterpoint book rules" in their full complexity; indeed the whole programming project was completed within three weeks, using a batch system. The same task would probably be far more difficult had we used traditional programming methods like nested if-then-else statements. As far as the rules are concerned, however, we did actually break a rule here and there, but on purpose, and for "musical" reasons (see the Appendix for a list of the rules we used).

As one would expect, we found out after our initial experiments that the entire set of "counterpoint book rules", coupled with blind search, were grossly insufficient for producing acceptable output. So we had to introduce "new" counterpoint rules. Here is one:

> Define a "high corner" to be a note that is preceded and followed by notes that are lower in pitch. E.g. in ...A B G... the B is a high corner (a local maximum). Rule: If a note occurs as a high corner, then it cannot occur as a high corner again for at least three measures. We define the same rule for "low corners" in an analogous manner. The melody is assumed to be preceded and followed by a note that is infinitely low in pitch (silence).

Here is another rule of our own:

> If the melody skips, and if the notes within the scope of this skip have not already been sounded, then they must eventually be sounded before the end of the melody.

We also imposed a search strategy that would help enumerate the more "musical" melodies first. According to this strategy, the program makes the following preferences when trying to append a new note to a partially completed melody:

> First, if there is a descending or ascending scale of crochets and quavers, try to continue that scale in the same direction.
>
> Second, try to proceed by step, giving preference to notes that have not already occured.
>
> Third, try to skip, giving preference to notes that have not already occured.

It is clear that there are many cases where this preference strategy will not be able to make a deterministic choice among a set of notes. The preference strategy has been programmed to make a random choice in such cases, which helps to ensure that the output of the program is not trivially predictable.

In general, the performance of the program seemed to compare quite well to that of an average "conservatory student", even for the "worst" outputs. Two examples taken from the output of the program have been given on the next two pages.

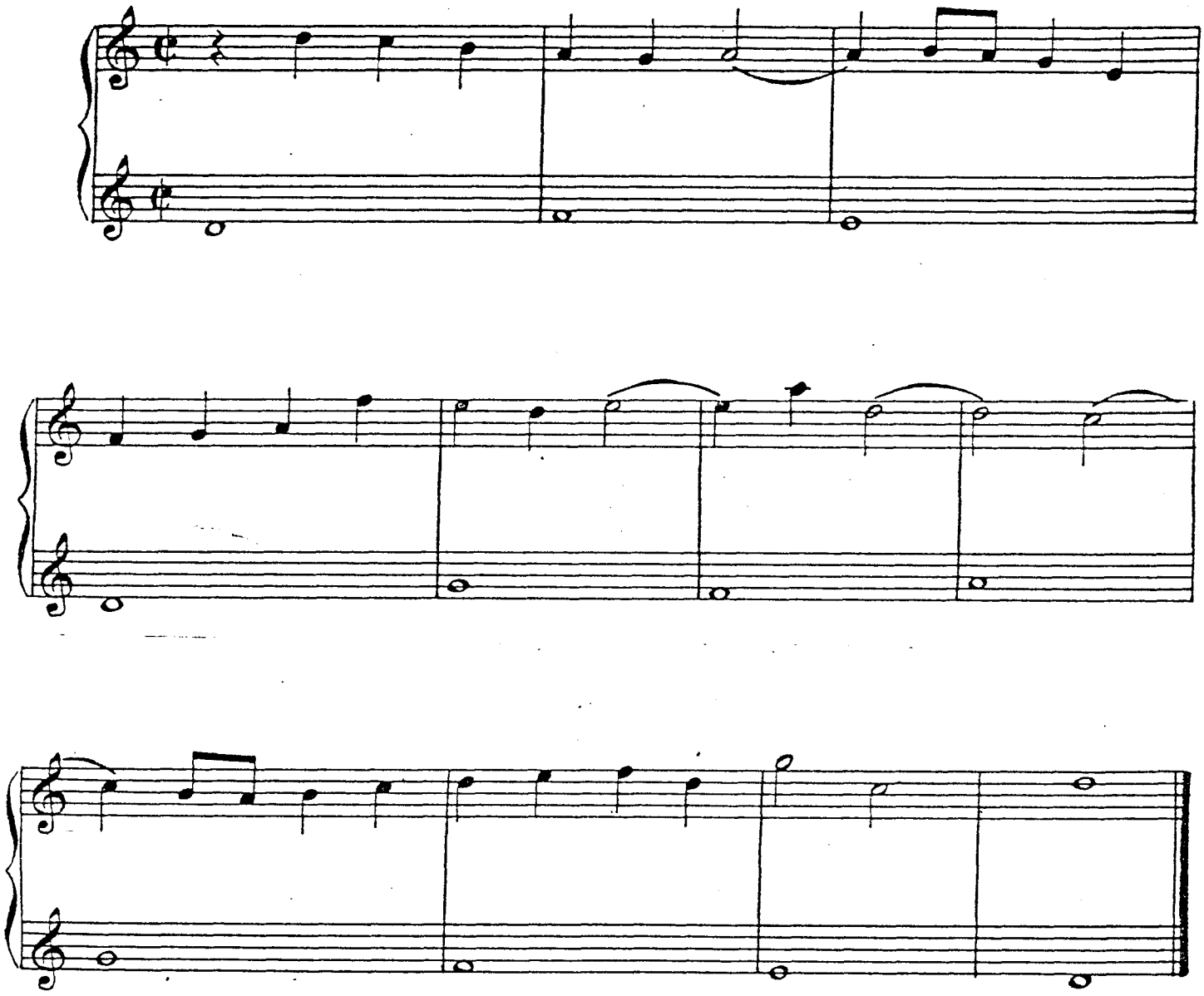
References:

[1] Koechlin, Ch. "Précis des Règles du Contrepoint", Editions Heugel, 1926.

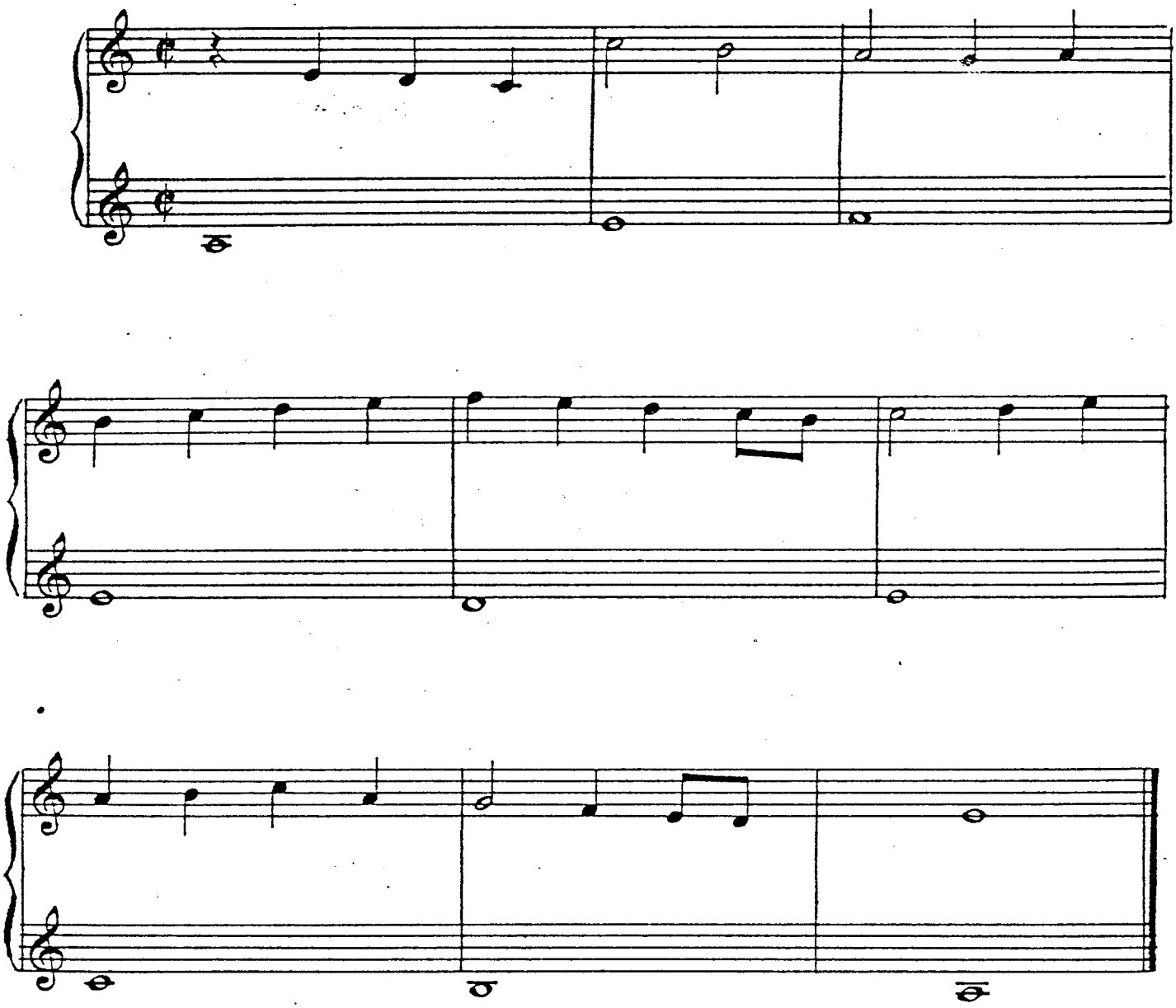
[2] Akses, N.K. Unpublished Lecture Notes in Strict Counterpoint (after Joseph Marx), Ankara State Conservatory, 1972-73. (Note that Joseph Marx was following J.J. Fux to a great extent, consequently, the influence of Fux is also visible in our program's rule structure)

[3] Ebcioğlu, K. "Strict Counterpoint: A Case Study in Musical Composition by Computers", M.S. Thesis (in English), Department of Computer Engineering, September 1979, Middle East Technical University, Ankara

LAST SOLUTION FOR THE CANTUS FIRMUS IN THE DORIAN MODE



LAST SOLUTION FOR THE CANTUS FIRMUS IN THE AEOLEAN MODE



APPENDIX

I. Normal Characteristic Functions

Normal characteristic functions are defined in Lisp metalanguage. But for the benefit of those readers who may be unfamiliar with Lisp, we give below an informal definition of normal characteristic functions in a more understandable (though informal) notation. For a thoroughly formal treatment of NCF's, and the other results mentioned in this paper, the reader is referred to [3].

Some introductory definitions: We notate a list as follows: (E1 E2 ... Ek) . E1 is the first element of this list, E2 is the second, and so on. The empty list is denoted by (). The function first(x) extracts the first element of the list x. The function rest(x) yields x with the first element removed.

Now, we define a normal characteristic function f to be a function of the following form:

```
f(x) = g(x, initial value of y)
where
g(x,y) = if x is empty then decide_empty(y)
         else if condition(first(x),y)
              then g(rest(x),update(first(x),y))
         else decide(first(x),y)
```

The second argument, y, of g is a list structure that is used for remembering relevant features of the part of the melody that g has already scanned. While some 'condition' is true g 'update's y according to the element currently scanned (i.e. first(x)), and then goes right to the next element (i.e. calls itself recursively with arg. 1 = rest(x)). If the 'condition' becomes false or if g hits the end of the list, either the predicate 'decide' or the predicate 'decide_empty' is invoked. In all cases, f returns a truth value ("yes" or "no"), which indicates whether or not f has accepted its argument x.

II. List of Rules Used in the Program

In order to make the rules clearer, we first need to indicate the internal representation of the Contrapunctus as it is used in the program. The following is the list representation of a Contrapunctus consisting of k notes:

((pitch_1 dur_1) (pitch_2 dur_2) ... (pitch_k dur_k))

pitch_i and dur_i (i.e., obviously, the pitch and duration of the ith note in the contrapunctus) assume integer values, for i=1,2,...,k. For pitches, we use 1 for middle C, 2 for D, and so on. (We use white keys only, taking advantage of the fact that most authors do not allow modulations in strict counterpoint.) A rest is denoted by a negative pitch large in magnitude. The duration of a note is measured in time units, where one time unit equals the duration of one quaver. Hence a minim has duration equal to 4, for example. If a note is tied to another note over the barline, then these two notes are represented as one single element in the list representation of the melody.

The rules used in the program are given in the list below. As mentioned in the main text, a few of the rules are "new", however, we must concede that we are still not entirely satisfied with them, despite the fact that they do help to get results, as evidenced by the examples. Describing any given style of music via a truly exact and unequivocal "Treatise" remains a difficult problem, and we now believe that an elegant solution probably involves rules that use deeper musical attributes of Schenkerian flavor, rather than the ad hoc restrictions exploited here.

No more than 20 time units of consecutive quavers or crochets are allowed. (1 time unit=1 quaver)
No more than 4 consecutive minims are allowed.
No more than 2 consecutive syncopations are allowed.
Mixture of minims and syncopations are not allowed for more than 4 measures.
('time' is an attribute of a note that indicates the number of time units elapsed since the beginning of the measure, e.g. if a note m is at the very beginning of a measure then time(m)=0)
If time=0 then only a crochet or a minim or a semibreve is allowed.
If time=2 or 6 then only a crochet or a quaver is allowed.
If time=3 or 7 then only a quaver is allowed.

If time=4 then only a crochet or a minim or a (minim tied to a minim) or a (minim tied to a crochet) is allowed.
In the first measure, if time=2 then a quaver is not allowed.
A note is a semibreve if and only if it occurs in the last measure.
Pairs of quavers must be seperated by at least two measures.
The rhythm (crochet crochet minim) or (crochet quaver quaver minim) is allowed only just before the last measure.
The melody begins with a rest.
The first non-rest note of the melody and the last note of melody are either an octave or a fifth above the Cantus Firmus. (Charles Koechlin occasionally advocates ending on the fifth as a "variante")
Let x,y,z be consecutive pitches. Then not(|x-y|>1 and |x-z|>1 and |y-z|>1) (disallows arpeggios, sevenths and ninths that are reached via two skips, but some good progressions are also rendered unacceptable)
If x1,x2,x3,x4 are consecutive pitches then x1=x3 implies not(x2=x4).
Ascending melodic intervals larger than a minor sixth are not allowed, with the exception of the ascending octave.
Descending melodic intervals larger than a fifth are not allowed.
Augmented and diminished intervals are not allowed.
A tritone in three notes is not allowed (e.g. B A F).
A tritone in four notes must be followed by step in the same direction.
An octave skip must be preceded and followed by notes within its scope.
If there is an ascending or descending scalar movement of quavers or crochets at the end of the measure, then the last note of the measure cannot skip in the same direction over the barline.
Quavers are always approached and left by stepwise motion.
No pitch can be repeated. (i.e. immediate repetition)
The melody cannot cross under the Cantus firmus part, (the C.F. appears in the lower part all the time) or go higher than a given limit.
The range of the melody cannot exceed a twelfth.
No more than three consecutive skips are allowed.
Three skips are allowed only if one of them is a third.

There can only be one climax (pitch peak) in the melody.
Rule about "high corners" (see main text).
Rule about "low corners" (see main text).
Rule about notes within the scope of a skip (see main text).
No exposed or consecutive <perfect fifth>'s are allowed, where <perfect fifth> indicates the class of a perfect fifth, a perfect twelfth, etc.
No exposed or consecutive <octave>'s are allowed, where <octave> denotes the class of a unisson, an octave, etc.
<octave>'s must be seperated by more than 1 measure, unless they are exactly 3/4 measures apart. (A curious German rule which does not differentiate between an harmonic note, a passing note etc. while making the decision to reject two octaves seperated by several notes) The same rule also applies to <perfect fifth>'s.
The distance between the melody and the Cantus Firmus cannot exceed a tenth.
The first crochet of a measure must be consonant with respect to the Cantus Firmus.
Minims must be consonant with respect to the Cantus Firmus.
If a minim is tied to a minim then the first note of the tied pair must be consonant with respect to the Cantus Firmus. The same rule applies to a minim tied to a crochet.
If the second note of the pair (minim tied to minim) or the pair (minim tied to crochet) is dissonant, a note a step below the pair must occur on the second beat of the measure containing the second note of the pair (assuming 2/2 time). Also if (minim tied to crochet) is followed by a crochet then this crochet must be consonant.
A dissonant note can only be approached and left by stepwise motion.
If a note is dissonant (when it starts to sound) then it must be a crochet or a quaver.
A unisson cannot occur on the first beat of the measure.
A unisson cannot be approached by step.