CHAPTER 7

CONDITIONAL SELECTION II: CUMULATIVE FEEDBACK

This Chapter returns to the topic of composing with statistical distributions. As in Chapter 5, we presume that we have some collection of items (notes, phrases) and that for each item we wish to select one out of a limited repertory of options (durations, pitches, registers) according to a prescribed distribution. We discussed methods in Chapter 5 which are highly effective in realizing prescribed statistical distributions over fixed frames. However, because these methods ~~select~~ arrange options in wholly random order, we have no way of insuring that individual options will be evenly spread throughout the frame. For example, assume a frame containing 10 notes and suppose that we wish to distribute one C, two F's, three E's and four D's over this frame. Figure 7-1 illustrates two possible arrangements, one of which is 'well-mixed', the other 'lumpy'. While the 'lumpy' sequence may have its charm, it is the 'well-mixed' sequence which most strongly projects the statistical nature of the material. The random methods of Chapter 5 are no more partial to one arrangement than the other.

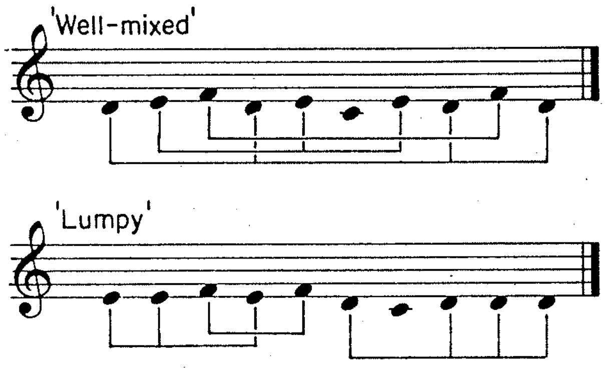


Fig 7-1

Figure 7-1: Two arrangements of a distribution.

Another problem with frame-dependent methods is that prior to any specific decision-making, it is necessary to determine how many items occur within a frame and perhaps also to total up durations over all these items. Sometimes this knowledge depends on the decisions themselves, so a priori statistical frames cannot be established.

Often one might wish to temper statistically based decisions with stylistic constraints or with practical criteria (such as whether something is playable). A third ~~problem~~ limitation with frame-dependent strategies is that the the available options narrow down until only one option remains for the last item in the frame. This feature renders frame-dependent strategies unamenable to constraints.

By contrast, cumulative feedback allows us to obtain 'well-mixed' distributions and to impose constraints while eliminating or at least substantially reducing dependence on frames. The process itself is very simple: each time it selects an an option, the composing program updates a cumulative statistic reflecting how much that particular option has been used. These statistics in turn enable the program to most greatly favor those options which lag farthest behind. Should one option be more heavily weighted than the others, the program compensates by advancing the heavily weighted option's statistic by a correspondingly smaller increments.

7.1 HEURISTIC SELECTION WITH CUMULATIVE FEEDBACK

Heurisms are bendable rules. More formally, they are procedures which tend to produce desirable solutions to problems but which do not guarantee that any specific standards will be met. Heurisms are often distinguished from algorithms, which -- in the strict sense of "algorithm" -- always produce correct solutions but which only work when such solutions exist. Heurisms, by contrast, are capable of providing solutions even under less-than-optimal conditions.

The most direct heurism to employ in cumulative feedback is the practice of always selecting the option which lags farthest behind its prescribed usage. This heurism was first used by the author for his composition Undulant (1983) as a means of guiding a constrained search (this concept is discussed in Chapter 14). It qualifies as a heurism because it is amenable to constraints (see Heading 7.~~4~~5); however, it also produces optimally 'well-mixed' distributions when no constraints are imposed.

7.1.1 Implementation

The library subroutine HEUR implements this heurism in an unconstrained context. HEUR requires seven arguments:

1. RESULT - HEUR selects an index I to one of NUM options stored in array VALUE and transfers VALUE(I) to RESULT (line 18). RESULT may be either an integer or a real number.

2. VALUE - Repertory of options. VALUE must be an array whose dimension in the calling program is NUM and whose type is identical to that of RESULT.

3. CUM - Cumulative statistics reflecting how much each option has previously been used. After it selects an index I, HEUR increases CUM(I) by DUR/WEIGHT(I) (line 20). CUM must be a real array of dimension NUM.

4. WEIGHT - Relative weights associated with each of the NUM options. WEIGHT must be a real array of dimension NUM whose elements are all greater than zero.

5. SCHED - This array provides the schedule by which HEUR

considers options. SCHED must be an integer array containing the values 1 through NUM.

6. DUR - Duration of item for which RESULT is being selected. DUR must be real.

7. NUM - Number of options. NUM must be an integer.

Notice that HEUR asks the library subroutine SHUFLE (heading 5.2) to permute SCHED randomly at the onset of each call (line 6); this step eliminates any bias between options with equal cumulative statistics.

If decisions are to reflect numbers of occurances rather than cumulative durations, simply set DUR consistently to 1.0. Graduated comprimises between numbers of occurances and cumulative durations may be obtained by raising the duration of an item to exponents ranging from 0.0 (numbers of occurances only) to 1.0 (cumulative durations only). It may also be desirable to increase DUR for items with non-durational accents such as sforzandi or extreme registers.

-- Programming example 7-1: subroutine HEUR --

Ex 7-1

```
 1          subroutine HEUR(RESULT,VALUE,CUM,WEIGHT,SCHED,DUR,NUM)
 2          dimension VALUE(1)
 3          real CUM(1),WEIGHT(1)
 4          integer SCHED(1)
 5          data HUGE/10000000.0/
 6          call SHUFLE(SCHED,NUM)
 7    C     Search for option with smallest cumulative statistic
 8          CMIN = HUGE
 9          do (J=1,NUM)
10            L = SCHED(J)
11            C = CUM(L)
12            if (C.lt.CMIN) then
13              CMIN = C
14              I = L
15            end if
16          repeat
17    C     Transfer this option to RESULT
18          RESULT = VALUE(I)
19    C     Update cumulative statistic
20          CUM(I) = CMIN + DUR/WEIGHT(I)
21          return
22          end
```

7.1.2 Example: Composing a 'Well-Mixed' Melody

We illustrate the strategy of heuristic selection by employing subroutine HEUR to compose a 'well-mixed' melody. As with the example used to illustrate Koenig's RATIO feature (heading 5.6.2), the composing process depicted in Figure 7-2 divides into three stages of production: I) periods between attacks, II) durations, and III) pitches.

Figure 7-2: Composing a 'well-mixed' melody - In Stage I, thin lines indicate periods of length two; medium lines indicate periods of length three; bold lines indicate periods of length five. In Stage II, phrase markings above noteheads indicate ties; markings below noteheads indicate slurs.

7.1.2.1 Stage I: Periods

The repertory of periods encompasses three options, with weights scaled in order to keep the cumulative amount of time occupied by each option in balance:

1. periods of length 2 receive weight 1/2,

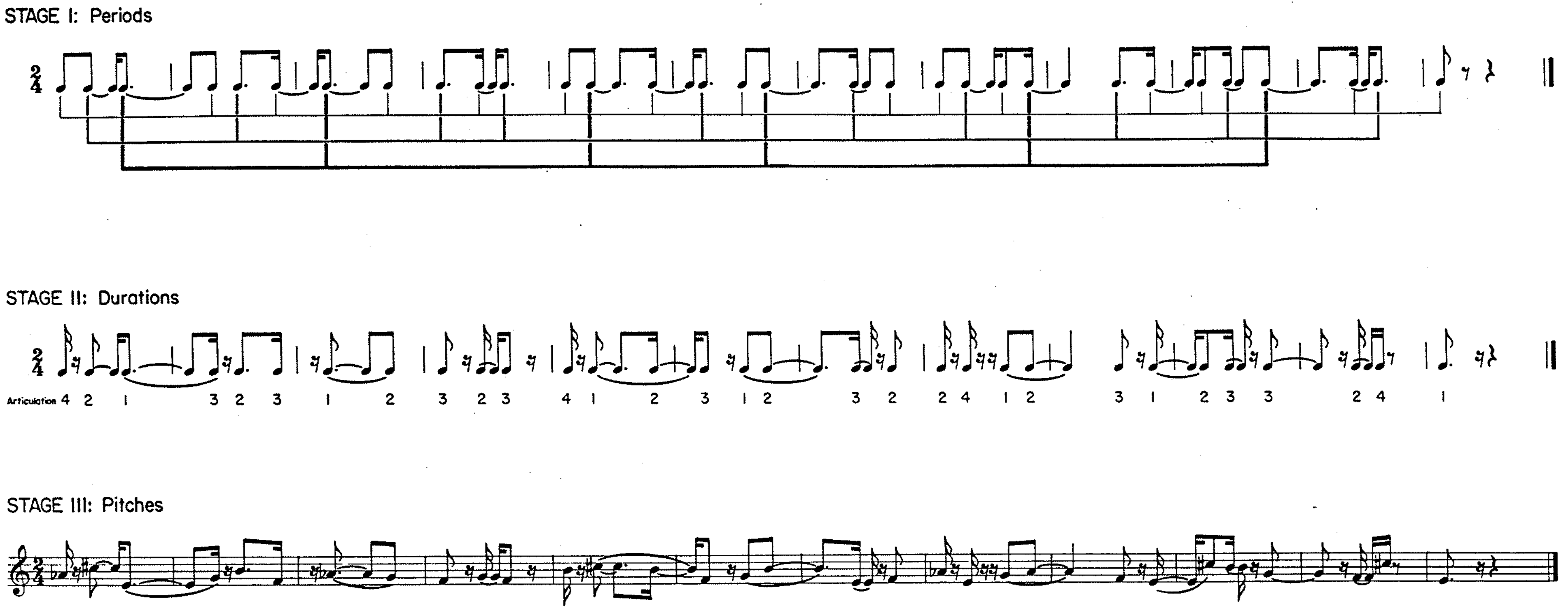

STAGE I: Periods
STAGE II: Durations
Articulation 4 2 1 3 2 3 1 2 3 2 3 4 1 2 3 1 2 3 2 2 4 1 2 3 1 2 3 3 2 4 1
STAGE III: Pitches

Fig 7-2

7-6a

2. periods of length 3 receive weight 1/3, and

3. periods of length 5 receive weight 1/5.

Periods are selected by numbers of occurances; the DUR argument to subroutine HEUR is set consistently to unity.

7.1.2.2 Stage II: Durations - In this stage, calls to subroutine HEUR return an articulation, which is in turn used to calculate a duration from the period. The repertory of articulations encompasses four options:

1. Slurred - With weight 2., the current note slurs to its successor. Duration is calculated as period plus one; for example, a slurred note with period 5 receives a duration of 6.

2. Sustained - With weight 3., the current note holds through its period, but the next note is tounged. Duration equals period; a sustained note with period 5 receives a duration of 5.

3. Detached - With weight 3., the current note falls short of its successor by a sixteenth rest. Duration is calculated as period minus one; a detached note with period 5 receives a duration of 4.

4. Short - With weight 1., the current note fills only half of its period. Duration is calculated by dividing the period by two and then rounding downward; a short note with period 5 receives a duration of 2.

Articulations, like periods, are selected relative to numbers of occurances.

7.1.2.3 Stage III: Pitches - HEUR selects pitches from a repertory of six equally weighted options: E4, F4, G4, Ab4, B4, and C#5. Selection of pitches is sensitive to cumulative duration; the duration calculated in Stage II provides the DUR argument to subroutine HEUR. Table 7-1 details the random schedules derived by HEUR for pitches through calls to the library subroutine SHUFLE along with the cumulative statistics computed for each pitch prior to each decision in Stage III.

Measure & Beat	Duration	Schedule and Cumulative Statistics						Selected Pitch
1:0	1.	Ab:0.	C#:0.	B:0.	F:0.	E:0.	G:0.	Ab4
1:2	3.	Ab:1.	C#:0.	E:0.	F:0.	G:0.	B:0.	C#5
1:5	6.	E:0.	F:0.	B:0.	G:0.	Ab:1.	C#:3.	E4
2:2	1.	E:6.	G:0.	C#:3.	B:0.	F:0.	Ab:1.	G4
2:4	3.	B:0.	F:0.	E:6.	G:1.	Ab:1.	C#:3.	B4
2:7	1.	C#:3.	B:3.	Ab:1.	F:0.	G:1.	E:6.	F4
3:1	6.	B:3.	Ab:1.	G:1.	E:6.	C#:3.	F:1.	Ab4
3:6	2.	E:6.	B:3.	G:1.	F:1.	Ab:7.	C#:3.	G4
4:0	2.	Ab:7.	G:3.	C#:3.	E:6.	F:1.	B:3.	F4
4:3	2.	G:3.	F:3.	E:6.	B:3.	C#:3.	Ab:7.	G4
4:5	2.	F:3.	G:5.	E:6.	C#:3.	B:3.	Ab:7.	F4
5:0	1.	Ab:7.	B:3.	C#:3.	E:6.	F:5.	G:5.	B4
5:2	6.	B:4.	G:5.	F:5.	Ab:7.	C#:3.	E:6.	C#5
5:7	2.	C#:9.	Ab:7.	E:6.	F:5.	G:5.	B:4.	B4
6:1	2.	C#:9.	F:5.	B:6.	E:6.	G:5.	Ab:7.	F4
6:4	3.	B:6.	Ab:7.	G:5.	E:6.	F:7.	C#:9.	G4
6:6	5.	F:7.	G:8.	B:6.	C#:9.	E:6.	Ab:7.	B4
7:3	2.	C#:9.	E:6.	B:11.	Ab:7.	F:7.	G:8.	E4
7:6	2.	C#:9.	G:8.	F:7.	Ab:7.	B:11.	E:8.	F4
8:0	1.	G:8.	F:9.	Ab:7.	C#:9.	E:8.	B:11	Ab4
8:2	1.	E:8.	C#:9.	F:9.	Ab:8.	B:11.	G:8.	E4
8:5	3.	G:8.	E:9.	Ab:8.	B:11.	F:9.	C#:9.	G4
8:7	5.	G:11.	Ab:8.	C#:9.	E:9.	F:9.	B:11	Ab4
9:4	2.	G:11.	B:11.	F:9.	C#:9.	E:9.	Ab:13	F4
9:7	3.	G:11.	F:11.	Ab:13.	E:9.	B:11.	C#:9.	E4
10:1	2.	E:12.	G:11.	B:11.	Ab:13.	F:11.	C#:9.	C#5
10:3	2.	B:11.	C#:11.	F:11.	Ab:13.	E:12.	G:11	B4
10:6	4.	G:11.	F:11.	Ab:13.	E:12.	B:13.	C#:11	G4
11:3	2.	F:11.	C#:11.	B:13.	Ab:13.	G:15.	E:12	F4
11:5	1.	G:15.	Ab:13.	F:13.	B:13.	C#:11.	E:12	C#5
12:0	3.	G:15.	F:13.	E:12.	Ab:13.	B:13.	C#:12	E4

Table 7-j

Table 7-1: Selection of pitches by subroutine HEUR.

7.2 RANDOM SELECTION WITH CUMULATIVE FEEDBACK

Figure 7-3: Mechanics of subroutine DECIDE.

We now consider a strategy for loosening up the heuristic procedures of the previous section in order to obtain shades of 'lumpiness' ranging along a continuum from the optimally 'well-mixed' sequences produced by subroutine HEUR to fully randomized sequences. Developed by the author, this strategy involves 1) deriving likelihoods of selection for each option based on how far an option lags behind the most-used option, and 2) applying the procedures of the library subroutine SELECT (heading 4.2.2.5) to make a decision. Figure 7-3 illustrates how likelihoods are derived. Each likelihood is the sum of two components:

1. A deviation reflects how much a specific option's cumulative usage falls short of the most-used option. The deviation associated with the most-used option is

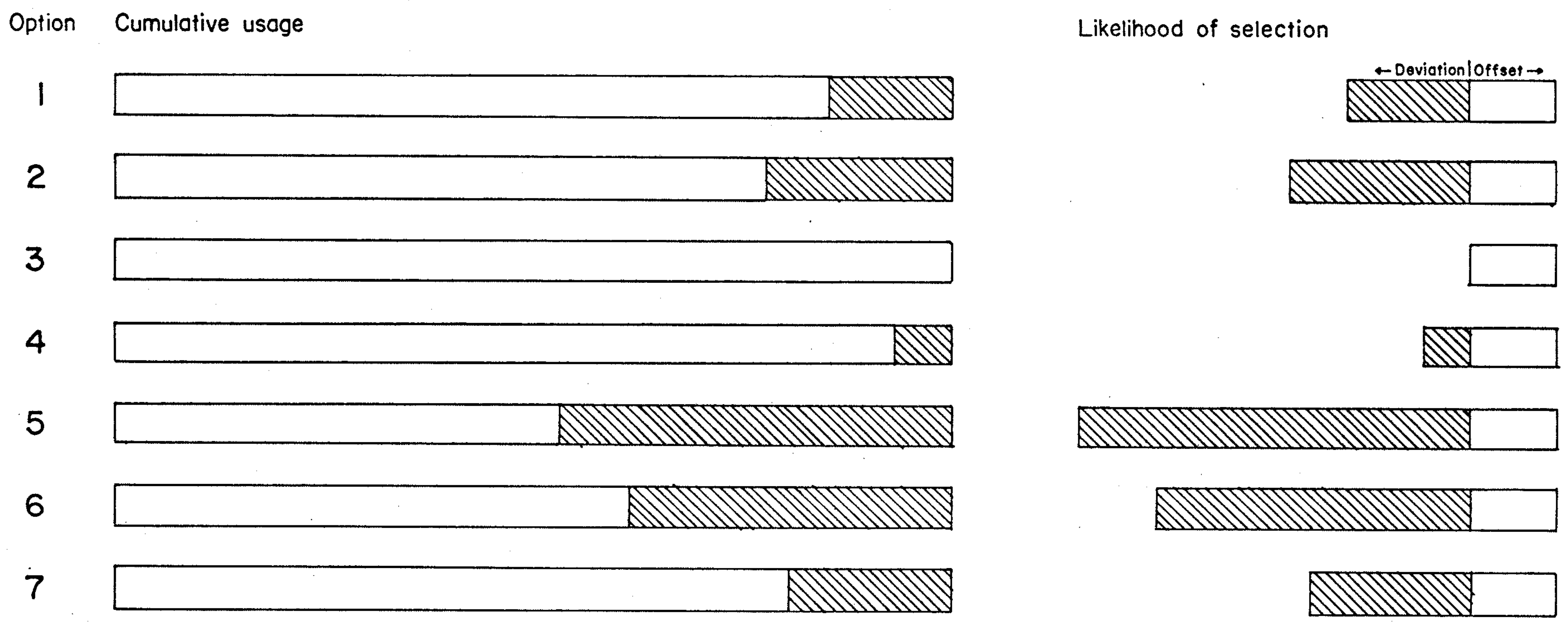

Option
Cumulative usage
Likelihood of selection
← Deviation | Offset →
1
2
3
4
5
6
7

Fig 7-3

always zero; the farther an option lags behind the most-used one, the greater its deviation and, hence, its likelihood of selection.

2. An <u>offset</u> reflects the degree of 'lumpiness'. The offset remains constant for all options; however, it most strongly affects the likelihood of selecting the most-used option. When the offset is zero, the strategy never selects the most-used option. As the offset grows, the strategy requires increasingly larger frames before the actual distribution of options comes to reflect the prescribed distribution.

7.2.1 Implementation

The library subroutine DECIDE implements this strategy. DECIDE requires seven arguments:

1. RESULT - DECIDE selects an index I to one of NUM values from array VALUE and transfers VALUE(I) to RESULT (line 19). RESULT may be either an integer or a real number.

2. VALUE - Repertory of options. VALUE must be an array of dimension NUM whose type is identical to that of RESULT.

3. CUM - Cumulative statistics reflecting how much each option has previously been used. After it selects an index I, DECIDE increases CUM(I) by DUR/WEIGHT(I) (line 21). CUM must be a real array of dimension NUM.

4. WEIGHT - Relative weights associated with each of the NUM options. WEIGHT must be a real array of dimension NUM whose elements are all greater than zero.

5. OFFSET - Likelihood of selection associated with the most-used option. OFFSET must be real.

6. DUR - Duration of item for which RESULT is being selected. DUR must be real.

7. NUM - Number of options. NUM must be an integer.

Numbers of occurances may be substituted for cumulative durations as the criterion of selection by setting DUR consistently to 1.0. Offsets should be proportional to the smallest potential increment occuring in line 20. To compute this smallest

Ex 7-2

```
      subroutine DECIDE(RESULT,VALUE,CUM,WEIGHT,OFFSET,DUR,NUM)
      dimension VALUE(1),CUM(1),WEIGHT(1)
C     Determine largest cumulative statistic
      T = 0.
      CMAX = 0.
      do (J=1,NUM)
        C = CUM(J)
        T = T + C
        CMAX = amax1(CMAX,C)
      repeat
C     Select option
      R = RANF() * (float(NUM)*(CMAX+OFFSET)-T)
      do (I=1,NUM)
        W = CMAX - CUM(I) + OFFSET
        if (R.le.W) exit
        R = R - W
      repeat
C     Transfer this option to RESULT
      RESULT = VALUE(I)
C     Update cumulative statistic
      CUM(I) = CUM(I) + DUR/WEIGHT(I)
      return
      end
```

increment, simply divide the smallest value expected for DUR by the largest value stored in WEIGHT.

-- Programming example 7-2: subroutine DECIDE --

7.2.2 Example: Composing a 'Lumpy' Melody

Figure 7-4: Composing a 'well-mixed' melody - In Stage I, thin lines indicate periods of length two; medium lines indicate periods of length three; bold lines indicate periods of length five. In Stage II, phrase markings above noteheads indicate ties; markings below noteheads indicate slurs.

We illustrate the strategy of random selection with cumulative feedback by employing subroutine DECIDE to compose a 'lumpy' melody. In order to compare this strategy directly to the procedure implemented in subroutine HEUR, we stipulate that the melody will be composed once again in three stages and the process be governed by exactly the same compositional data

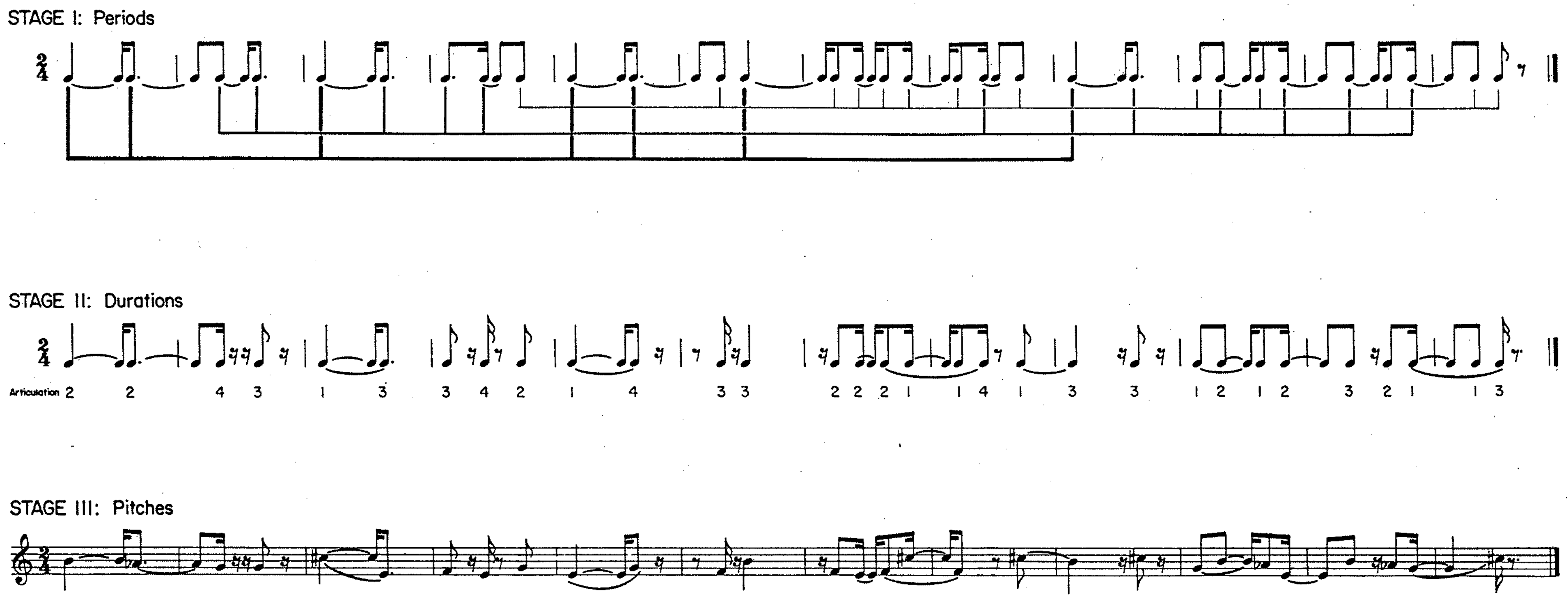

STAGE I: Periods
STAGE II: Durations
Articulation 2 2 4 3 1 3 3 4 2 1 4 3 3 2 2 2 1 1 4 1 3 3 1 2 1 2 3 2 1 1 3
STAGE III: Pitches

Fig 7-4

described under heading 7.1.2. Figure 7-4 illustrates the stages of production. The 'lumpiness' of this new melody is due to the offsets provided to DECIDE, which have been computed as follows:

Stage I: Periods - Since periods are selected relative to numbers of occurances, DUR will always be 1.0. The largest weight for any period is 1/2 (the weight for periods of length 2), so the smallest potential increment is 2.0. The offset used to select the periods depicted in Figure 7-4 is twice this smallest increment, or 4.0.

Stage II: Durations - Articulations are also selected relative to numbers of occurances, so DUR will again always be 1.0. The largest weight for any articulation is 3.0 (the weights for sustained and detached articulations) so the smallest potential increment is 1/3. The offset used to select the articulations depicted in Figure 7-4 is three times this smallest increment, or 1.0.

Stage III: Pitches - Durations resulting from the articulations selected in Stage II can range down to a minimum of one sixteenth note. The weights for all pitches are uniformly 1.0, so the smallest possible increment is also 1.0. The offset used to select the pitches depicted in

Measure & Beat	Duration	Cumulative Statistics and Weights E4	F4	G4	Ab4	B4	C#5	Selected Pitch
1:0	5.	0. (2.)	0. (2.)	0. (2.)	0. (2.)	0. (2.)	0. (2.)	B4
1:5	5.	0. (7.)	0. (7.)	0. (7.)	0. (7.)	5. (2.)	0. (7.)	Ab4
2:2	1.	0. (7.)	0. (7.)	0. (7.)	5. (2.)	5. (2.)	0. (7.)	G4
2:5	2.	0. (7.)	0. (7.)	1. (6.)	5. (2.)	5. (2.)	0. (7.)	G4
3:0	6.	0. (7.)	0. (7.)	3. (4.)	5. (2.)	5. (2.)	0. (7.)	C#5
3:5	3.	0. (8.)	0. (8.)	3. (5.)	5. (3.)	5. (3.)	6. (2.)	E4
4:0	2.	3. (5.)	0. (8.)	3. (5.)	5. (3.)	5. (3.)	6. (2.)	F4
4:3	1.	3. (5.)	2. (6.)	3. (5.)	5. (3.)	5. (3.)	6. (2.)	E4
4:6	2.	4. (4.)	2. (6.)	3. (5.)	5. (3.)	5. (3.)	6. (2.)	G4
5:0	6.	4. (4.)	2. (6.)	5. (3.)	5. (3.)	5. (3.)	6. (2.)	E4
5:5	2.	10. (2.)	2. (10.)	5. (7.)	5. (7.)	5. (7.)	6. (6.)	G4
6:2	1.	10. (2.)	2. (10.)	7. (5.)	5. (7.)	5. (7.)	6. (6.)	F4
6:4	4.	10. (2.)	3. (9.)	7. (5.)	5. (7.)	5. (7.)	6. (6.)	B4
7:1	2.	10. (2.)	3. (9.)	7. (5.)	5. (7.)	9. (3.)	6. (6.)	F4
7:3	2.	10. (2.)	5. (7.)	7. (5.)	5. (7.)	9. (3.)	6. (6.)	E4
7:5	2.	12. (2.)	5. (9.)	7. (7.)	5. (9.)	9. (5.)	6. (8.)	F4
7:7	3.	12. (2.)	7. (7.)	7. (7.)	5. (9.)	9. (5.)	6. (8.)	C#5
8:1	3.	12. (2.)	7. (7.)	7. (7.)	5. (9.)	9. (5.)	9. (5.)	F4
8:3	1.	12. (2.)	10. (4.)	7. (7.)	5. (9.)	9. (5.)	9. (5.)	F4
8:6	3.	12. (2.)	11. (3.)	7. (7.)	5. (9.)	9. (5.)	9. (5.)	C#5
9:0	4.	12. (2.)	11. (3.)	7. (7.)	5. (9.)	9. (5.)	12. (2.)	B4
9:5	2.	12. (3.)	11. (4.)	7. (7.)	5. (10.)	13. (2.)	12. (3.)	C#5
10:0	3.	12. (4.)	11. (5.)	7. (9.)	5. (11.)	13. (3.)	14. (2.)	G4
10:2	3.	12. (4.)	11. (5.)	10. (6.)	5. (11.)	13. (3.)	14. (2.)	F4
10:5	3.	12. (4.)	14. (2.)	10. (6.)	5. (11.)	13. (3.)	14. (2.)	Ab4
10:7	3.	12. (4.)	14. (2.)	10. (6.)	8. (8.)	13. (3.)	14. (2.)	E4
11:2	2.	15. (2.)	14. (3.)	10. (7.)	8. (9.)	13. (4.)	14. (3.)	B4
11:5	2.	15. (2.)	14. (3.)	10. (7.)	8. (9.)	15. (2.)	14. (3.)	Ab4
11:7	4.	15. (2.)	14. (3.)	10. (7.)	10. (7.)	15. (2.)	14. (3.)	G4
12:2	3.	15. (2.)	14. (3.)	14. (3.)	10. (7.)	15. (2.)	14. (3.)	G4
12:4	1.	15. (4.)	14. (5.)	17. (2.)	10. (9.)	15. (4.)	14. (5.)	C#4

Table 7-2

Figure 7-4 is twice this smallest increment, or 2.0.

Table 7-2 details the cumulative statistics and weights derived by subroutine DECIDE for each pitch prior to each decision in Stage III.

Table 7-2: Selection of pitches by subroutine DECIDE.

7.3 'PRIMING' A DISTRIBUTION

HEUR and DECIDE both associate equal likelihoods to each option at the outset: only after an option has been selected for the first time does its relative weight come to influence decisions. This condition results simply because all cumulative statistics initially begin at zero; in no way does it comprimise the overall distribution of options. One may prod HEUR or DECIDE to give a more representative distribution at the outset by determining the average duration AVGDUR and then initializing each CUM(I) to AVGDUR/WEIGHT(I) as in the following except of FORTRAN '77 code. However, this action does nothing to prevent the elements of CUM to come into alignment later in the decision-making process.

Ex 7-3

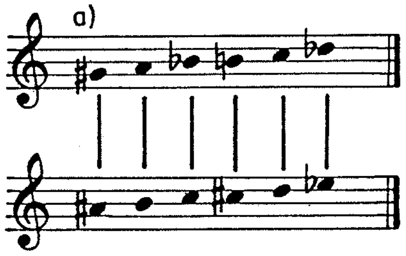
```
do (I=1,NUM)
  CUM(I) = AVGDUR/WEIGHT(I)
repeat
```

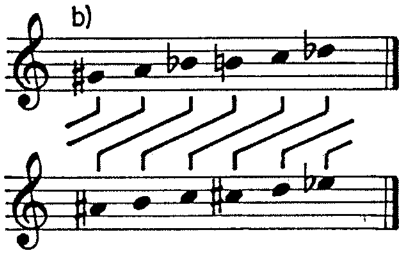
-- Programming example 7-3 --

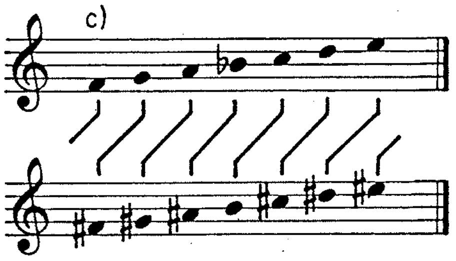
7.4 MODULATION

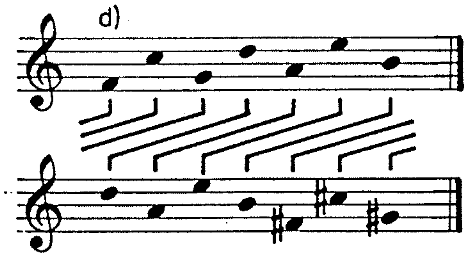
In frame-dependent strategies we would expect changes in repertory to coincide with the beginning of a new frame, where -- by the nature of the strategy -- all old options will have been selected according to their prescribed distribution. The situation differs for the frame-independent strategies discussed in this chapter, which might have allowed one or more old options to fall considerably behind due to random trends or to constraints. Resetting cumulative statistics to zero restarts the process with a clean slate, wiping out all imperative to restore such lagging options into balance. When both the old and new repertories have the same number of options, an alternative is to transfer the cumulative statistic for each old option to a direct counterpart or proxy in the new repertory.

7.4.1 Mappings

a)


b)


c)


d)


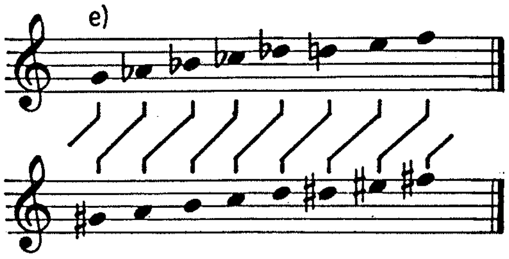
e)


Fig 7-5

Figures 7-5, a to e, illustrate changes in repertories of chromatic degrees. The bold lines linking pairs of degrees in each Figure indicate potential mappings by which statistics accumulated for one repertory may be transferred to its successor.

Figure 7-5: Modulatory mappings.

One obvious mapping might be to transfer statistics between degrees with like indices. This approach would seem to make special sense in Figure 7-5a, where the new repertory can be regarded as a transposition of the old repertory. However, it poses serious perceptual problems. Assume for simplicity that the relative weights associated with each degree are uniform. Suppose also that owing to randomness or to constraints in the previous music, B has fallen behind its prescribed density, while Db (later C#) has gotten ahead of itself. From the listener's perspective, another Db would seem redundant, yet the mapping in Figure 7-5b would encourage selection of precisely this degree (as C#), which at the moment of modulation assumes the lagging statistic formerly associated with B. A more effective mapping from the listener's perspective would be the one depicted ~~that illustrated~~ in Figure 7-5b, which transfers cumulative statistics directly between identical degrees (or enharmonic equivalents). One should not regard Figure 7-5b's mappings between non-idential degrees as implying any equivalence of musical function beyond the fact that G# and D are each excluded from the other's repertory, as are A and Eb. If the decision-making process has slighted Bb, B, C, and Db in favor of G# and A under the old repertory, then the criterion of statistical uniformity would suggest that A#, B, C, and C# receive their due in future decisions.

Other mappings are less problematic than those of Figures 7-5a and 7-5b. Figures 7-5c and 7-5d illustrate mappings between diatonic scales, one in stepwise representation (Figure 7-5c), the other in circle-of-fifths representation (Figure 7-5d). Figure 7-5e illustrates mappings between "octatonic" scales. Those degrees in each of these three cases which do not find direct counterparts in the new scale parts at least have proxies a semitone lower or higher. The mappings illustrated in Figures 7-5b through 7-5e all involve <u>rotating</u> cumulative statisitics until the old values line up with the new options. This process requires a procedure for rotating an array, such as the library subroutine ROTATE (heading 3.5.4).

-- Programming example 7-4: subroutine ALIGN --

The library subroutine ALIGN implements mappings for

Ex 7-4

```
      subroutine ALIGN(CUM,IOLD,INEW,MIDX,NUM)
      dimension CUM(1)
      if (1.gt.IOLD.or.IOLD.gt.MIDX .or. 1.gt.INEW.or.INEW.gt.MIDX) then
        stop 'Bad argument to ALIGN.'
      end if
C     Determine 'nearest' distance between old and new indicies
      M = MIDX / 2
      ISHIFT = INEW - IOLD
      if (ISHIFT.le.-M) then
        ISHIFT = ISHIFT + MIDX
      else if (ISHIFT.ge.M) then
        ISHIFT = ISHIFT - MIDX
      end if
C     Rotate cumulative statistics
      call ROTATE(CUM,ISHIFT,NUM)
      return
      end
```

repertories such as those depicted in Figures 7-5b and 7-5d which exploit adjacent positions in the chromatic circle or in the circle of fifths. ALIGN requires five arguments:

1. CUM - Cumulative statistics. CUM must be a real array of dimension NUM in the calling program.

2. IOLD - Old index. IOLD must be an integer in the range from 1 to MIDX.

3. INEW - New index. INEW must be an integer in the range from 1 to MIDX.

4. MIDX - Maximum index. If ALIGN is used to map subsets of the chromatic circle or the circle of fifths, then MIDX will be 12. MIDX must be an integer.

5. NUM - Number of options.

For example, we might index each repertory of degrees depicted in Figure 7-5b by the leftmost degree, so that the first repertory would have index 9 while its successor would have index 11. If array CUM is a real array holding cumulative statistics for all six options, then the following call to ALIGN would rotate CUM

Ex 7-5

```
      call ALIGN(CUM,9,11,12,6)
```

Ex 7-6

```
      call ALIGN(CUM,11,2,12,7)
```

Ex 7-7

```
      do (ISEG=1,MSEG)
C       Align cumulative statistics with current options
        ...

C       'Sensitize' cumulative statistics to current weights
        do (I=1,NUM)
          CUM(I) = CUM(I) + AVGDUR(ISEG)/WEIGHT(I,ISEG)
        repeat
C
C       Compose segment
        ...

C       'Desensitize' cumulative statistics to current weights
        do (I=1,NUM)
          CUM(I) = CUM(I) - AVGDUR(ISEG)/WEIGHT(I,ISEG)
        repeat
      repeat
```

the number of shifts for the modulation depicted in Figure 7-5b.

-- Programming example 7-5 --

Similarly, we might index each repertory of degrees depicted in Figure 7-5d by the position of the leftmost degree on the circle of fifths illustrated in Figure 1-2b. F resides at 11 o'clock on this circle while D resides at 2 o'clock. If array CUM is a real array holding cumulative statistics for all seven options, then the following call to ALIGN would rotate CUM the number of shifts for the modulation depicted in Figure 7-5d.

-- Programming example 7-6 --

7.4.2 'Priming' New Segments

When the relative weights are not uniform, one will often desire the most highly weighted options to receive greatest emphasis at the beginning of a new repertory. Here is a strategy for acheiving this goal. Let AVGDUR(ISEG) designate the average duration during the segment of music during which the ISEGth repertory holds sway. Let NUM designate the number of options in each repertory (NUM may not change from segment to segment), let CUM(I) hold a cumulative statistic and let WEIGHT(I,ISEG) hold the weight associated with the Ith option during the ISEGth segment. Then the following excerpt of FORTRAN '77 code acts to 'prime' cumulative statistics so that choices will reflect the new distribution at the onset of each new segment:

-- Programming example 7-7 --

7.5 CONSTRAINTS

The two strategies described for frame-independent selection with cumulative feedback, heuristic and random, are both amenable to the imposition of constraints. Should a constraint force either of these strategies to select a statistically inferior option during a given decision, feedback allows the strategy to compensate for this choice in later decisions. Because constraints typically reflect very local relationships while statistical considerations reflect the broad scope, the two approaches to decision-making tend to complement each other rather than competing.

Since constraints are invariably idiosyncratic, it is not feasible to implement the mechanism of selection within a closed subroutine. For heuristic selection, one may employ the code of the library subroutine HEUR almost entirely, with the exception that the conditional test (line 12) should also test for acceptability. For random selection, it is necessary to compute likelihoods of selection for each option subject to conditional tests imposing likelihoods of zero upon those options found to be unacceptable. Examples of both procedures occur in program DEMO5, described under the next heading.

7.6 DEMONSTRATION 5: CUMULATIVE FEEDBACK

Demonstration 5 illustrates how cumulative feedback might be used in a composing program. It employs the structure of phrases and notes used in Demonstrations 2 and 3 with the enhancement that phrases are distinguished not only by phrase lengths, average durations, articulations, and registers, but also by scales.

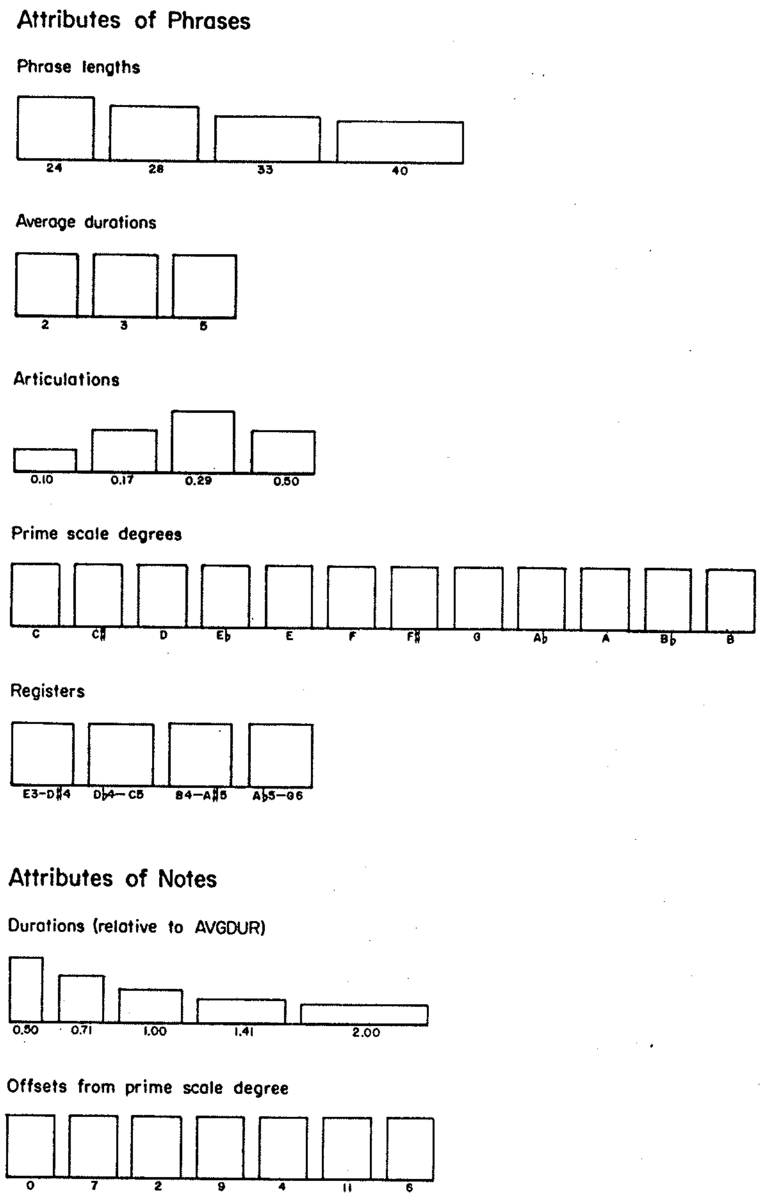

Attributes of Phrases
Phrase lengths
24 28 33 40
Average durations
2 3 5
Articulations
0.10 0.17 0.29 0.50
Prime scale degrees
C C♯ D E♭ E F F♯ G A♭ A B♭ B
Registers
E3–D♯4 D♭4–C5 B4–A♯5 A♭5–G6
Attributes of Notes
Durations (relative to AVGDUR)
0.50 0.71 1.00 1.41 2.00
Offsets from prime scale degree
0 7 2 9 4 11 6

Fig 7-6

7.6.1 Compositional Directives

Figure 7-6: Compositional data for Demonstration 5.

Attributes of phrases are selected randomly with cumulative feedback in accordance with the statistical distributions depicted in Figure 7-6. Selection of phrase durations, average durations, articulations and registers is unconstrained; selection of prime scale degrees is subject to two provisions: 1) no two consecutive scales may share more than four common degrees, and 2) no two consecutive primes may be related by a tritone.

Although The basic form of a~~ny given~~ scale corresponds to the "lydian mode", ~~although it~~ it is not used as such because the primary serves simply as an index for the program -- not as a reference for the listener. For our purposes, it is most useful to think of the basic form as the collection of degrees generated by a sequence of six rising perfect fifths above the primary.

Contrast between individual transpositions of this basic form results from cross relations (note 1). The number of cross relations occuring between any pair of transpositions depends on the nearest distance between their primaries around the circle of fifths illustrated in Figure 1-2a. For examples, Figure 7-7 shows that a scale starting on C has one cross relation and six

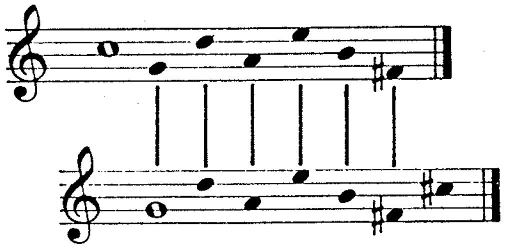
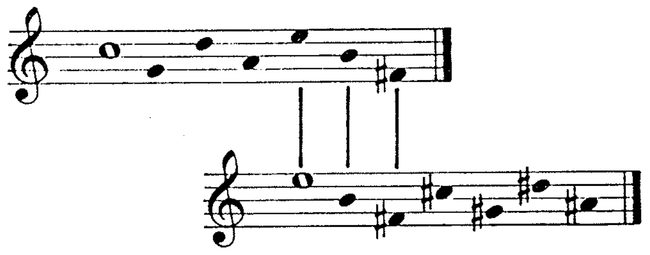


Fig 7-6



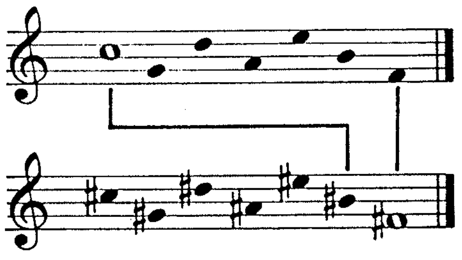
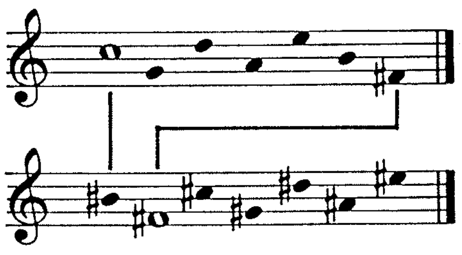


Fig 7-7



common degrees with a scale starting on G, but a scale starting on C has four cross relations and three common degrees with a scale starting on E.

Figure 7-7: Common-tone relationships.

The prohibition against tritone-related primaries is pragmatic, motivated by the ambivalent common tones resulting when two scales are related by a tritone. Notice in Figure 7-8 how aligning the two F#'s throws C out of alignment with B#, while aligning C against B# throws the F#'s out:

Figure 7-8: An uncooperative relationship.

Figure 7-9: Stylistic matrix for Demonstration 5.

Also depicted in Figure 7-6 are the exponential distribution used to select durations for notes (rests last half as long) and the uniform distribution used to select degrees. The last attribute is selected heuristically with sensitivity to duration. It is also constrained by the stylistic matrix depicted in Figure 7-9. This matrix rejects any two consecutive degrees related by a tritone, any three consecutive notes containing two identical degrees, and any three consecutive notes whose degrees make up

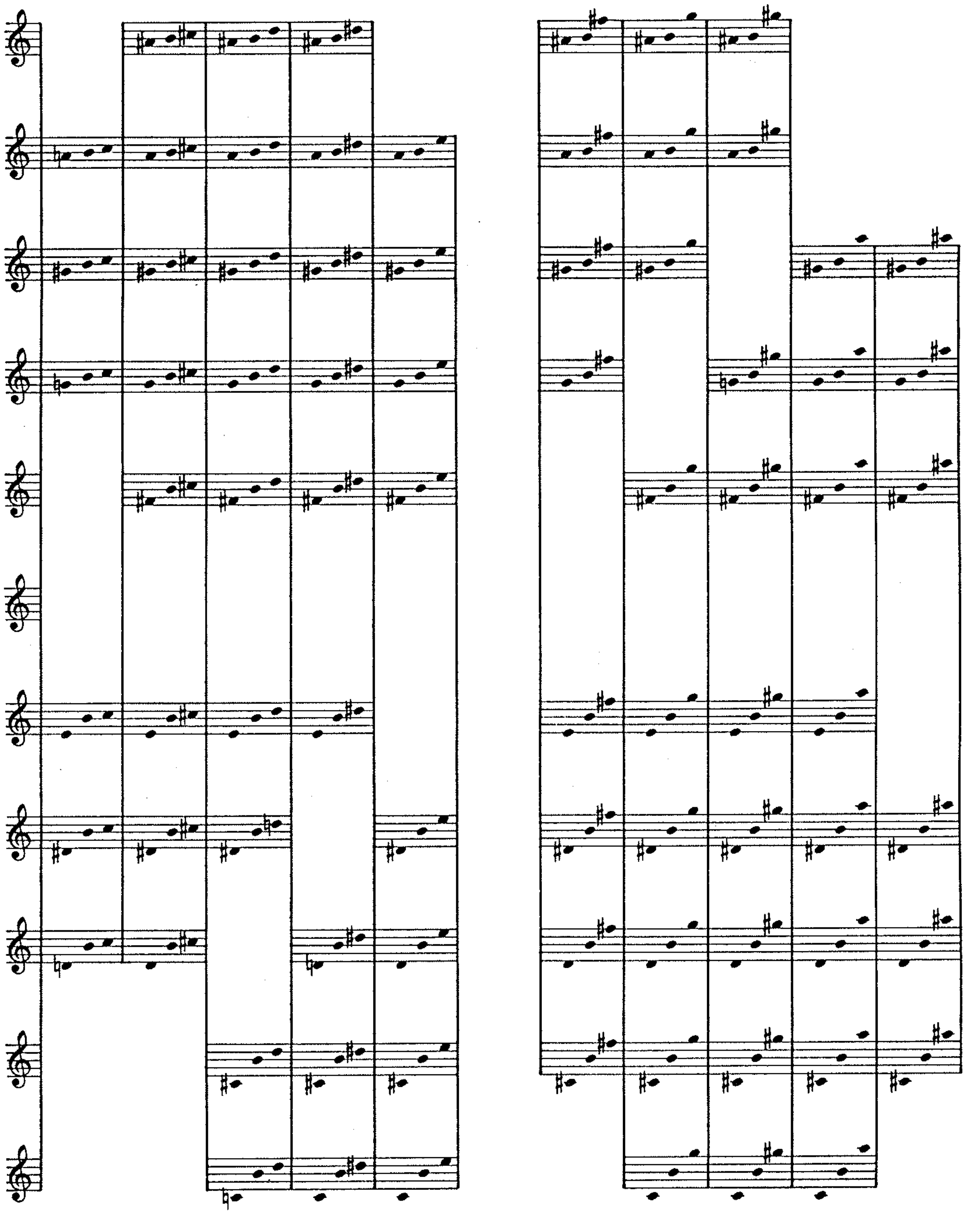
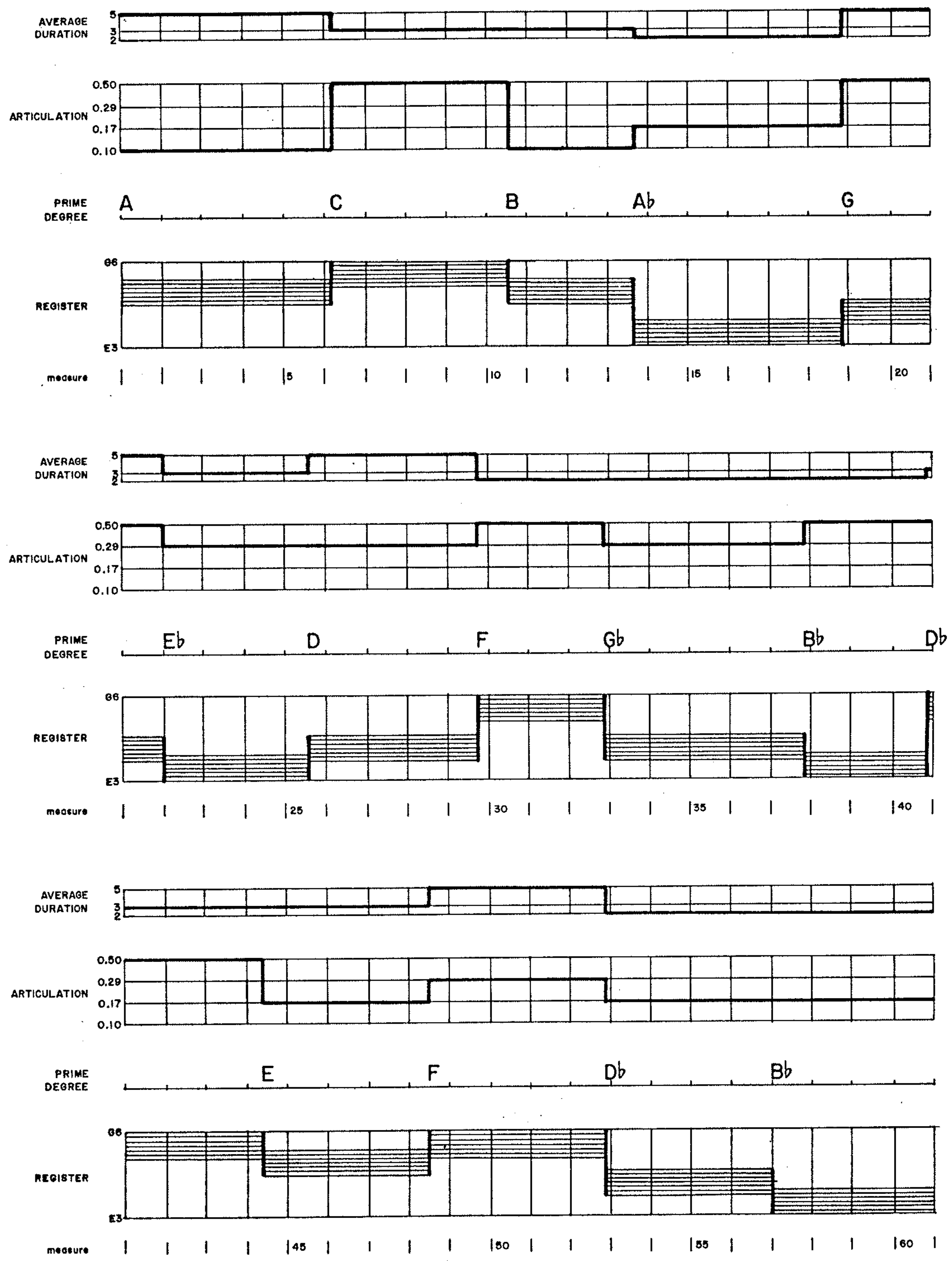


Fig 7-9



AVERAGE DURATION
5
3
2
ARTICULATION
0.50
0.29
0.17
0.10
PRIME DEGREE
A
C
B
Ab
G
REGISTER
G6
E3
measure
5
10
15
20
Eb
D
F
Gb
Bb
Db
25
30
35
40
E
F
Db
Bb
45
50
55
60

Fig 7-10

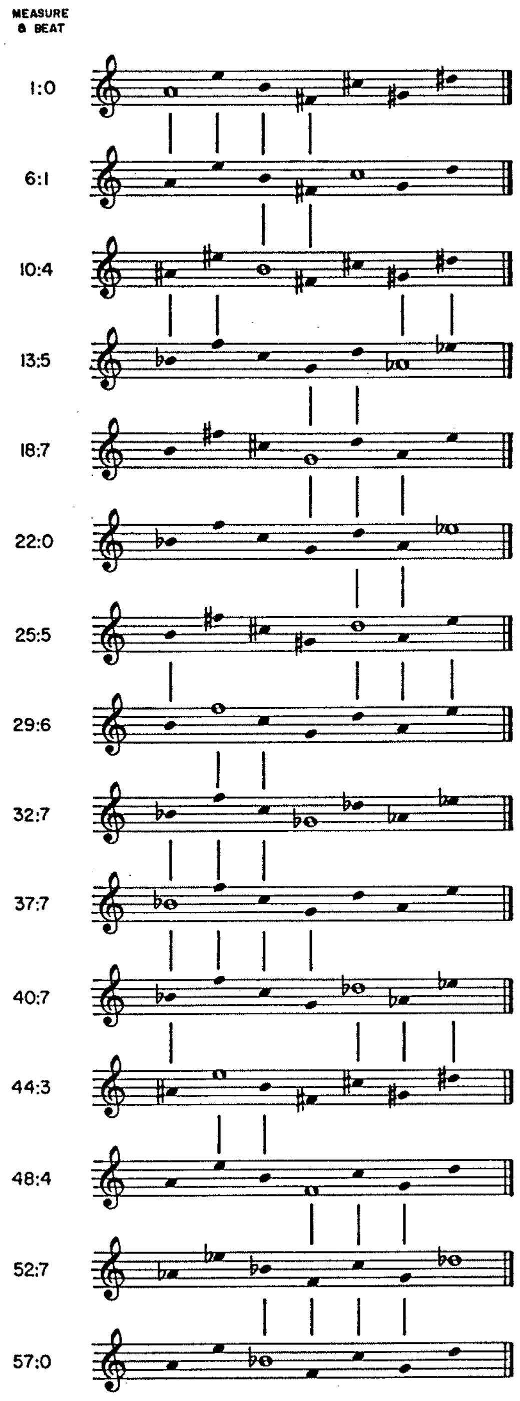

MEASURE & BEAT
1:0
6:1
10:4
13:5
18:7
22:0
25:5
29:6
32:7
37:7
40:7
44:3
48:4
52:7
57:0

Fig 7-11

7-23d

Demonstration 5

Clarinet

STRICTLY ♩ = 80

Charles AMES

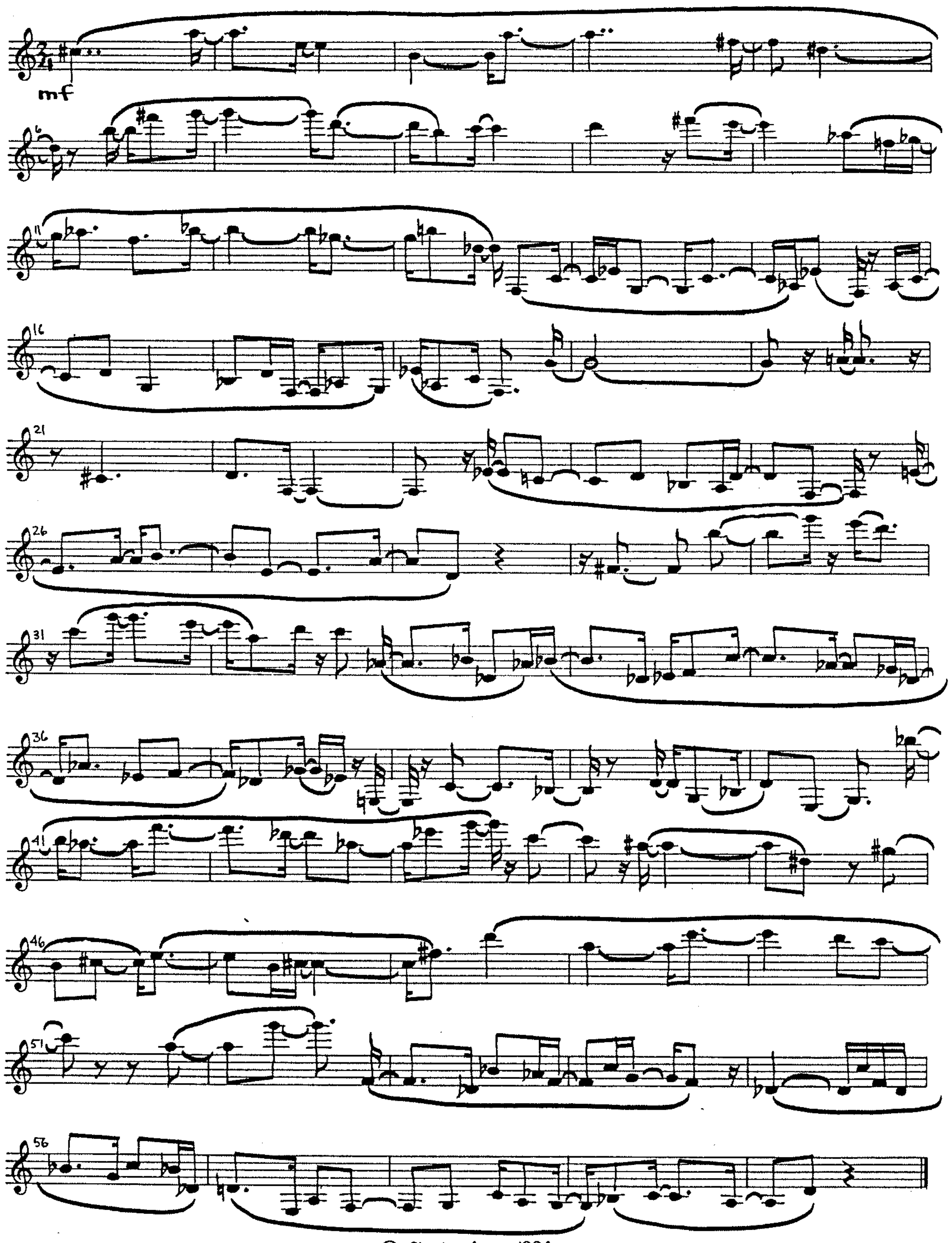


Fig 7-12

dissonant triads of the forms C-C#-D, C-F#-B, or C-F-B, in any inversion, voicing, or transposition. The stylistic matrix works along with the intervallic relationships imposed by the scales -- which each consist of seven adjacent positions on the circle of fifths -- to establish a noticably more consonant style than is apparent in Demonstration 4.

Figure 7-10 graphs the musical attributes selected by the composing program for phrases; Figure 7-11 details the progression of scales and their linkages through common tones. The complete product appears in Figure 7-12.

Figure 7-10: Profile of Demonstration 5.

Figure 7-11: Progression of scales in Demonstration 5 - Whole-note heads signify 'prime' degrees; vertical lines indicate degrees shared between consecutive scales.

Figure 7-12: Transcription of Demonstration 5.

7.6.2 Implementation

```
      program DEMO5
C
C     Demonstration of cumulative feedback
C
      parameter (MPHR=4,MAVG=3,MART=4,MREG=4,MPRM=12)
      integer VALREG(MREG),VALPHR(MPHR),VALPRM(MPRM)
      integer SCDPRM(MPRM)
      real VALAVG(MAVG),VALART(MART)
      real WGTPHR(MPHR),WGTAVG(MAVG),WGTART(MART),
     :     WGTREG(MREG)
      real CUMPHR(MPHR),CUMAVG(MAVG),CUMART(MART),
     :     CUMREG(MREG),CUMPRM(MPRM)
      common KTIME,ITIME,AVGDUR,ARTIC,IREG,IPRM,IDXPRM,IDXPR1
      data VALPHR/ 24, 28, 33, 40/,WGTPHR/.30,.26,.22,.19/,
     :     CUMPHR/0., 0., 0., 0./
      data VALAVG/2.,3.,5./,WGTAVG/1.,1.,1./,
     :     CUMAVG/0.,0.,0./
      data VALART/.10,.17,.29,.5/,WGTART/1.,2.,3.,2./,
     :     CUMART/0.,0.,0.,0./
      data VALREG/40,49,59,68/,WGTREG/1.,1.,1.,1./,
     :     CUMREG/0.,0.,0.,0./
      data VALPRM/1,8,3,10,5,12,7,2,9,4,11,6/,
     :     SCDPRM/1,2,3,4,5,6,7,8,9,10,11,12/,
     :     CUMPRM/0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,0./
      data HUGE/10000000.0/
C
C     Initialize
C
      open (2,file='DEMO5.DAT',status='NEW')
      ITIME = 0
      MTIME = 8 * 60
      IDXPRM = IRND(MPRM)
C
C     Outer composing loop
C
      do
C       Select duration of phrase
        call DECIDE(IPHR,VALPHR,CUMPHR,WGTPHR,3.3,1.0,MPHR)
        KTIME = KTIME + IPHR
C       Test for end of composition
        if (KTIME.gt.MTIME) exit
        PHR = float(IPHR)
C       Select average duration for notes in phrase
        call DECIDE(AVGDUR,VALAVG,CUMAVG,WGTAVG,4.0,PHR,MAVG)
C       Select probability of rest
        call DECIDE(ARTIC,VALART,CUMART,WGTART,4.0,PHR,MART)
C       Select register
        call DECIDE(IREG,VALREG,CUMREG,WGTREG,4.0,PHR,MREG)
C       Select primary scale degree
        call SHUFLE(SCDPRM,MPRM)
        IDXPR1 = IDXPRM
        CMIN = HUGE
        do (J=1,MPRM)
          I = SCDPRM(J)
          INTRVL = iabs(I-IDXPR1)
          if (min0(INTRVL,12-INTRVL).gt.2 .and. INTRVL.ne.6) then
            C = CUMPRM(I)
            if (C.lt.CMIN) then
              CMIN = C
              IDXPRM = I
            end if
          end if
        repeat
        IPRM = VALPRM(IDXPRM)
        CUMPRM(IDXPRM) = CMIN + PHR
C       Compose phrase
        call PHRASE
      repeat
      close (2)
      stop
      end

      subroutine PHRASE
      parameter (MDEG=7,MTVL=11,MDUR=5)
      integer VALDEG(MDEG),IGLTVL(MTVL,MTVL)
      real VALDUR(MDUR),CUMDEG(MDEG),CUMDUR(MDUR),WGTDEG(MDEG),
     :     WGTDUR(MDUR)
      logical SUCCES,REST,LGLTVL(MTVL,MTVL)
      common KTIME,ITIME,AVGDUR,ARTIC,IREG,IPRM,IDXPRM,IDXPR1
      equivalence (LGLTVL,IGLTVL)
      data VALDEG/0,7,2,9,4,11,6/,CUMDEG/0.,0.,0.,0.,0.,0.,0./
      data VALDUR/0.5,0.71,1.0,1.41,2.0/,WGTDUR/2.0,1.41,1.0,0.71,0.5/
     :     CUMDEG/0.,0.,0.,0.,0./
      data IDEG/1/,ITVL/7/,REST/.true./
```

Ex 7-8

```
      data IGLTVL/ 0,-1,-1,-1, 0, 0,-1,-1,-1, 0, 0,
     :            -1,-1,-1,-1,-1, 0,-1,-1,-1, 0, 0,
     :            -1,-1,-1,-1,-1, 0,-1,-1, 0,-1,-1,
     :            -1,-1,-1,-1,-1, 0,-1, 0,-1,-1,-1,
     :             0,-1,-1,-1,-1, 0, 0,-1,-1,-1,-1,
     :             0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     :            -1,-1,-1,-1, 0, 0,-1,-1,-1,-1, 0,
     :            -1,-1,-1, 0,-1, 0,-1,-1,-1,-1,-1,
     :            -1,-1, 0,-1,-1, 0,-1,-1,-1,-1,-1,
     :             0, 0,-1,-1,-1, 0,-1,-1,-1,-1,-1,
     :             0, 0,-1,-1,-1, 0, 0,-1,-1,-1, 0/
C
C     Inner composing loop
C
      call ALIGN(CUMDEG,IDXPR1,IDXPRM,12,MDEG)
      do
C       Select note or rest (no two consecutive rests)
        if ( REST .or. .not.SUCCES(ARTIC) ) then
C         Select duration of note
          call DECIDE(DUR,VALDUR,CUMDUR,WGTDUR,0.25,1.0,MDUR)
          DUR = AVGDUR*DUR + REMAIN
          IDUR = max0(1,ifix(DUR+0.5))
          REMAIN = DUR - float(IDUR)
C         Determine longest cumulative usage of any scale degree
          CMAX = 0.
          do (I=1,MDEG)
            C = CUMDEG(I)
            if (CMAX.lt.C) CMAX = C
          repeat
C         Compute weights for each scale degree
          SUM = 0.
          do (I=1,MDEG)
            LDEG = IPRM + VALDEG(I)
            if (LDEG.gt.12) LDEG = LDEG - 12
            LTVL = LDEG - IDEG
            if (LTVL.lt.0) LTVL = LTVL + 12
            if (LTVL.gt.0 .and. LGLTVL(LTVL,ITVL)) then
C             Weight for acceptable degree is deviation from longest
C             cumulative usage offset by 2.
              W = CMAX - CUMDEG(I) + 2.0
            else
C             Weight for unacceptable degree is zero
              W = 0.
            end if
            WGTDEG(I) = W
            SUM = SUM + W
          repeat
C         Select scale degree
          R = RANF() * SUM
          do (I=1,MDEG)
            W = WGTDEG(I)
            if (R.le.W) exit
            R = R - W
          repeat
          LDEG = IDEG
          IDEG = IPRM + VALDEG(I)
          if (IDEG.gt.12) IDEG = IDEG - 12
          ITVL = IDEG - LDEG
          if (ITVL.lt.0) ITVL = ITVL + 12
          CUMDEG(I) = CUMDEG(I) + float(IDUR)
C         Write note
          call WNOTE(ITIME,IDUR,IDEG,IREG)
          REST = .false.
        else
C         Select duration of rest
          call DECIDE(DUR,VALDUR,CUMDUR,WGTDUR,0.25,1.0,MDUR)
          DUR = AVGDUR/2.0*DUR + REMAIN
          IDUR = ifix(DUR)
          REMAIN = DUR - float(IDUR)
C         Write rest
          call WNOTE(ITIME,IDUR,0,0)
          REST = .true.
        end if
C       Test for end of phrase
        if (ITIME.ge.KTIME) exit
      repeat
      return
      end
```

-- Programming example 7-8: program DEMO5 (2 pages) --

Like programs DEMO2 and DEMO3, program DEMO5 implements the musical structure through an "outer" composing loop (lines 36-68 of DEMO5 proper) for phrases and an "inner" composing loop (lines 28-88 of subroutine PHRASE) for notes and rests. The symbols of DEMO5 adhere to six mnemonic 'roots' corresponding to attributes of phrases:

1. PHR - length of phrase.

2. AVG - average duration of notes (equivalently, average tempo); the average duration of rests is half as large.

3. ART - articulation, expressed as the probability that a rhythmic unit will serve as a rest.

4. PRM - primary scale degree for a phrase

5. REG - register, expressed as the lowest pitch in a one-octave gamut.

In addition, subroutine PHRASE includes three additional mnemonic roots corresponding to attributes of notes:

1. DEG - degree of a seven-note scale whose transposition is determined by IPRM.

2. DUR - duration.

3. TVL - interval in semitones.

The number of options available to each attribute is given by a parameter starting with the letter M. The values reside in arrays beginning with VAL, their associated weights reside in arrays beginning with WGT, and cumulative statistics in arrays beginning with CUM.

Because selection of phrase lengths reflects numbers of occurances and because the largest weight appearing in array WGTPHR is .30, the smallest increment occuring in DECIDE's line 21 will be 3.333. The offset used to select phrase lengths approximately equals this smallest increment.

Selection of average durations, articulations, and registers all reflect lengths of phrases. Since shortest phrase length is always 24, while the largest weights stored in arrays WGTAVG and WGTREG are both 1.0, the smallest increment in DECIDE's line 21 will be 24.0 in each case. The offset of 4.0 used to select average durations and registers is a sixth of this amount, which produces 'well-mixed' choices. The smallest increment for articulations is 24/3, or 8.0, so articulations are somewhat 'lumpier'.

Each scale used in Demonstration 5 is a transposition of the basic form stored in array VALDEG (line 10 of subroutine PHRASE; each degree is expressed as an offset giving the number of semitones above the primary degree IPRM). Lines 50-64 of program DEMO5 use the methods of the library subroutine HEUR to select a primary degree in such a manner as to provide equal durational emphasis to each of the twelve primaries subject to the constraints described earlier.

7.6.2.1 Rhythm - PHRASE asks the library subroutine DECIDE to select durations of notes and rests (lines 42-45 for notes, 88-91 for rests) in an approach similar to that of Demonstration 3. Here, however, the repertory of durations is restricted to five values clustered around the average duration. Since durations are selected with sensitivity to numbers of occurances and the largest weight in array WGTDUR (line 10) is 2.0, the smallest increment used in DECIDE's line 21 will be 0.5. The offset for both notes and rests is half of this smallest increment.

7.6.2.2 Pitches - A call to the library subroutine ALIGN prior to entering the "inner" loop (line 27 of subroutine PHRASE) insures that degrees shared in common by consecutive scales will retain their cumulative statistics. PHRASE selects scale degrees for notes using the strategy of the library subroutine DECIDE, but imposes stylistic constraints upon consecutive intervals. PHRASE applies these constraints as it computes weights for degrees (lines 55-68); degrees found to violate a constraint receive null weights. The constraints themselves include a rule forbidding identical degrees for two consecutive notes along with the stylistic matrix detailed in Figure 7-9. Information describing this matrix is stored as the 11x11 logical matrix LGLTVL (lines 13-23 of PHRASE).

7.7 NOTES

1. This attitude toward cross relations derives from Arnold Schoenberg's Preliminary Exercises in Counterpoint, 1963.