PITCH-CLASS MULTISETS

by

THOMAS ROBINSON

A dissertation submitted to the Graduate Faculty in Music in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

2009

This manuscript has been read and accepted for the
Graduate Faculty in Music in satisfaction of the dissertation
requirement for the degree of Doctor of Philosophy.

Jonathan Pieslak

Date — Chair of Examining Committee

David Olan

Date — Executive Officer

Phillip Lambert

Shaugn O'Donnell

Joseph Straus

Abstract

PITCH-CLASS MULTISETS

by

Thomas Robinson

Advisor: Professor Joseph Straus

The *pitch-class multiset* (pcmset) is a collection in which pitch classes may appear as elements more than once and in which any single appearance of a pc represents one and only one instance of that pitch class. For example, pitch classes 1, 2, and 4 comprise the pcmset {1,2,4,4}; pc4 occurs twice. This represents some musical situation with two instances of pc4 and only one instance each of pcs 2 and 3. The pcmset has appeared sporadically in the theoretical literature, yet there has been no systematic examination into the ramifications of the distinction between a pitch class and the number of its representatives. This study considers existing music theory in light of pcmsets and considers their use in analysis.

First, from an ontological perspective, this study carefully defines the pcmset as distinct from the pitch set and the pitch-class set. Once the relationship between the canonical set classes and *multiset classes* is established, what follows is an expansive, combinatorial survey of thousands of mset classes.

Second, this study revisits the standard tools and concepts of pc-set theory. The interval-class vector, the Z-relation, and complementation all are modified only minimally to accommodate pcmsets and mset classes. What is more, this accommodation gives new insight into the nature of these principles.

Throughout, this study uses pcmsets in music analysis by identifying parent class and pcmsets in Webern's Opus 5, by looking at their Fourier balance in a Bach chorale, and by tracking transformations of pitch-class multiplicity in the music of Arvo Pärt.

ACKNOWLEDGEMENTS

I extend maximal gratitude to my advisor, mentor, and friend Joseph Straus. As any good advisor would do, he confidently led me from the dissertation's inception—the moment he told me what a multiset is—to its completion. Moreover, he made everything seem like my idea all along. But he also did what only great mentors do—he demonstrated how to be a student and how to be a teacher. He showed me how to write, and how to speak, about music theory. He even told me how to get a job. The most valuable of his lessons, though, was usually more implicit than explicit, and it was simply this: whenever possible, talk less and listen more. Good advice. Joe's example is priceless, and it will stay with me for a long time.

Dissertations—good ones—require rigorous critique. Fortunately, mine benefitted from the comments of many obliging professionals. When I drifted toward the fantastical my reader, Shaugn O'Donnell, was right there with some reality checks, but he never delivered them without a reassuring chuckle. Philip Lambert and Jonathan Pieslak, brought their experience and expertise to the table, leaving this paper far superior to the one they first encountered. I am grateful to have them on my committee. For the conversation and correspondence that affected this dissertation, whether incidentally or profoundly, I'd like to thank Robert Morris, Stephen Peles, Ian Quinn, and Dmitri Tymoczko. Special thanks go to Richard Hermann, who opened the door in 2003.

I also thank my parents, who alternately supported and needled in just the right measure. However, the most gratitude is owed to the one who bore the burden of my deadlines missed, who endured the vacations that never were, and who repeatedly wiped away my doubt. Whatever success is found herein, I owe it all to my wife, Julie.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER ONE: PRELIMINARIES

1.1. Types, Tokens, and Occurrences

Philosophers and metaphysicians—as well as linguists, phoneticians, psychologists, and aestheticians—frequently make the useful distinction between a *type* and a *token*, a distinction similar but not identical to the more familiar one between universals and particulars. Tokens are the concrete instances of abstract entities, or types. Linda Wetzel elaborates:

> Consider the Grizzly (or Brown) Bear, *Ursus arctos horribilis*. At one time its U.S. Range was most of the area west of the Missouri River and it numbered 10,000 in California alone. Today its range is Montana, Wyoming and Idaho, and it numbers less than 1,000. Of course, no particular bear numbers 1,000 and no particular bear ever had a range comprising most of the area west of the Missouri. It is a *type* of bear, a species of bear, that has both properties.[1]

The Grizzly is a type, and the tokens are the many living, breathing creatures roaming the American West.

For our musical purposes, let "Middle C" be a type. One can find tokens of this type by sitting at a piano, pressing a particular key situated near the middle of the keyboard, and listening to the results. The first two measures of Chopin's *Ballade,* Op. 38 contain a total of seven tokens of this type, Middle C, whether they are heard in

[1] Linda Wetzel, "On Types and Words," *Journal of Philosophical Research* 27 (2002): 240. My conception of type, token, and occurrence was informed strongly by and primarily based on Wetzel's work. Also see her "What Are Occurrences of Expressions?" *Journal of Philosophical Logic* 22 (1993): 215-220 and *Types and Tokens: An Essay on Universals* (Cambridge, MA: M.I.T. Press, 2008).

concert performance or merely viewed as notations on the page. The sounds and the ink both form concrete tokens, or examples, of the abstract entity Middle C.[2]

A proper explication of multiset theory (MST) as distinct from classical set theory (CST) relies both on the type/token distinction and on a third ontological term, *occurrence*. In this section, I will apply the token, type, and occurrence to musical phenomena, but not before investigating the terms themselves.

1.1.1 Types and Tokens

Linguistics is an ideal territory in which to study types and tokens. Discussions of tokens and types usually name C.S. Peirce as the coiner of the terms:

> There will ordinarily be about twenty *the*'s on the page, and of course they count as twenty words. In another sense of the word "word," however, there is but one word "the" in the English language; and it is impossible that this word should lie visibly on a page or be heard in any voice, for the reason that it is not a Single thing or Single event. It does not exist; it only determines things that do exist. Such a definitely significant Form, I propose to term a *Type*. A Single event which happens once and whose identity is limited to that one happening or a Single object or thing which is in some single place at any one instant of time, such event or thing being significant only as occurring just when and where it does such as this word or that word on a single line of a single page of a single copy of a book, I will venture to call a *Token*.[3]

In it this excerpt there are 167 words or *word-tokens*. One arrives at this number by starting at the beginning and counting each group of letters separated by a blank space. Noticeably, words like *the*, *of*, and *single* appear more than once. Such repetition of word-tokens brings the number of unique words, or *word-types*, in the passage to 82.

---

[2] Of course, the *Ballade* (the entire piece) can be understood as a type of which there are many tokens, i.e. recordings, performances, analyses. Such tokens may even be "imperfect" instances of the work (type). Nicholas Wolterstorff refers to such works as *norm-types*: "[I]t is possible for them to have properly formed and also possible for them to have improperly formed examples." *Worlds and Works of Art* (Oxford: Clarendon Press, 1980), 56.

[3] C.S. Peirce, *Collected Papers of Charles S. Peirce*, Vol. IV The Simplest Mathematics, ed. Charles Hart Shorne and Paul Weiss (Cambridge, MA: Harvard University Press, 1933), 423. Emphasis original.

There are, then, roughly twice as many tokens in the excerpt as there are types.[4] Questions will arise even in a counting procedure as simple as this. Should the word-tokens *the* and *the's* be counted as the same word-type? Yes, I say, since the latter is the plural of the former. What about the word-tokens *happens* and *happening*? Maybe they are different types because in the excerpt the former is a verb while the latter is a noun. Maybe they do belong to the same type for they have the same root. These varied answers create uncertainty about what types really are. After all, another reader/counter might not even consider both *the* and *the's* to be tokens of the same type. One has three letters; the other has four plus an apostrophe. Clearly, they are inscriptions of different designs, but in an important sense both tokens refer to the same definite article. If my copy of Peirce's work had a rip or tear where the *t* should be (like so: •*he*) would the token's remains still constitute a token of the type *the*, or would it belong to the type *he*? Surely, someone reading my damaged copy likely would, without trouble, call to mind the abstract word-type *the* when arriving at the token •*he*.

Word-types not only can be instanced as tokens on the page but also can be vocalized as spoken tokens. However, similar ambiguities arise.

> In thick guttural accents one utters the utterance "I *v*ant a banana": what has he said? He has said this: "I *v*ant a banana." In reporting what is said in this sense one apes the speaker, emulates a tape recorder.
>
> One who utters the utterance "I *v*ant a banana" can also be said to have said "I want a banana."[5]

---

[4] These two data can be placed in a token/type ratio of 167:82, or 2.04. The closer this number is to 1, the richer one might deem vocabulary of the excerpt. The ratio (and its inverse, the type/token ratio) is also used to study the development of children's vocabulary but has some problems with regard to relative sizes of excerpts. Naturally, the longer the excerpt, the greater the author's (or speaker's) necessity to resort to word-types previously used. See G. Herdan, *Type-Token Mathematics: A Textbook of Mathematical Linguistics* (Mouton: The Hague, 1960).

[5] Paul Ziff, "What is Said," in *Semantics of Natural Language,* 2nd ed., ed. Donald Davidson and Gilbert Harman (Dordrecht, Holland: D. Reidel, 1972), 709-721.

With this example, Paul Ziff shows that all spoken tokens of a type need not be pronounced identically. That a multitude of accents and dialects can coexist within a single language without destroying its comprehensibility is evidence of speakers' reliance on word-types, abstract entities that exist prior to their spoken instantiation.[6]

Not only are there various ways to pronounce or to spell a single word-type, but there are also single spellings or single pronunciations that can refer to multiple types, i.e. homonyms and homophones. Surely, when Peirce wrote "lie visibly on a page," the inscription *page* refers to a "leaf in a manuscript," not a "youth being trained for the medieval rank of knight."[7] Likewise, if one were dictating Peirce's line "There will ordinarily be about twenty *the*'s on the page…," a stenographer should have enough sense not to write, "*They're* will ordinarily *bee* about twenty *these* on the page…" This suggests that tokens do not "find their own way" to their types. Nor do types automatically generate tokens. There is a certain human intervention, which, whether by intuition or logical argument, helps to make ontological decisions regarding the sorting of tokens into types.[8]

---

[6] Nominalists tend to deny the existence of such entities, while still acknowledging speakers' common reliance on them. Most nominalists would deny that any universals—even linguistic *types*—exist, reducing to a predicate any particular's reference to such a universal. See note 15 below. Linda Wetzel, in "The Trouble with Nominalism," *Philosophical Studies* 98 (2000): 361-70, takes on the nominalism of W.V. Quine and Nelson Goodman's "Steps Toward a Constructive Nominalism," in Goodman, *Problems and Projects* (New York: Bobbs-Merrill, 1972) and Goodman's *The Structure of Appearance,* 3rd Ed. (Dordrecht: D. Reidel, 1977).

[7] *Merriam-Webster Online Dictionary*, s.v. "page," http://www.merriam-webster.com/dictionary/page (accessed June 2008).

[8] This is where the token/type distinction parts ways with the universal/particular distinction. Richard Wollheim argues that "[a] very important set of circumstances in which we postulate types…is where we can correlate a class of particulars with a piece of human invention: these particulars may then be regarded as tokens of a certain type." *Art and Its Objects, 2nd Ed.* (Cambridge: Cambridge University Press, 1980[1968], 78. By modifying invention to in*ter*vention here gives more emphasis both to the sorting process and to humans' active role in it.

1.1.2 Which Tokens of Which Type?

Linda Wetzel poses a series of questions about (word-)types and (word-)tokens, one of which gets right to the heart of the matter: "Is there anything all and only tokens of a particular word have in common other than being tokens of that word?"[9] Is there some feature that, when identified, fixes the word as a token of a particular type? As seen in the cases of "th•" and "*v*ant" above, we can rely neither on spelling nor pronunciation of word-types to be this feature. What about mere similarity to other tokens? The spoken "*v*ant" indeed sounds *close* to the pronunciation of "want." Unfortunately, it is equally close to "vent." Similarity is even less sufficient for single letter-types and letter tokens. Wetzel, after Peter Ludlow, shows how different fonts often may obscure distinctions between tokens. For example, 8 is perhaps more similar (visibly) to 6 than to 8, but 8 and 6 are tokens of different types, while 8 and 8 are tokens of the same type.[10]

Critics may seize upon the vagueness of token-sorting criteria, denying abstract types altogether, but Wetzel pushes further. Could *intention* (of the speaker or writer) be the deciding criterion? In other words, can we look beyond accents, mispronunciations, and marred print? Not so in all cases, she writes. Even if there is a high correlation between intention and result, intention is neither necessary nor sufficient. "There is also a high degree of correlation between table forks and intentions to produce table forks. But it would be putting the cart before the horse to analyze forks, or words, in terms of intentions to produce them. Any account of what an intention-to-say-"cat" is will probably presuppose some account of what the word "cat" is."[11]

---

[9] Wetzel, "On Types," 242.
[10] Ibid., 245.
[11] Ibid., 248.

Ultimately she concludes that "*there is nothing that all and only tokens of a particular word have in common other than being tokens of that word*."[12] This leaves nothing inherent in a token to fix it as one type or another. To make the decision, some outside source is required, something referred to above as human intervention. Reluctant to consider types to be *natural* kinds—the term not is not only controversial but also problematic because word-tokens, produced by humans, "are not 'natural' but artifacts"[13]—Wetzel prefers the term *real* types.[14] Opting for *real* instead of *natural* also seems to harmonize with Wetzel's common-sense reference to the "type-talk" that surrounds us. Whether or not abstract types actually exist, language users act as if they do. In this sense, these types are real kinds, and it is sensible to deal with them as such.[15]

Biological taxonomy becomes a robust analogue for Wetzel's real kinds. First she tackles four species-sorting methods and shows how each is problematic. Then, for word-types, such methods are shown to be problematic in the very same ways.

1) Resemblance (Darwin)

Wetzel: A set of properties characterizes a biological species, but not all tokens have all characteristics, and some characteristics are held by tokens of other species. The same is true for word-tokens.

2) Same Genetic Code (Putnam)

---

[12] Ibid., 248. Emphasis original.
[13] Ibid., 249.
[14] Wetzel attributes an earlier, similar claim to Sylvain Bromberger, who proposed that "[t]okens of a type make up a quasi-natural kind," and who also went further to suggest that "their type is the archetype of that quasi-natural kind." See "Types and Tokens in Linguistics" in *On What We Know We Don't Know* (Chicago: University of Chicago Press, 1992), 176.
[15] One nominalist objection would be that these types are a sort of illusory shorthand. If we refer to the Ford Mustang, the nominalist might say that we are using shorthand either for "every Ford Mustang" or "every auto created by the Ford Motor Co. and given the moniker "Mustang." From this point of view there is no abstract Ford Mustang separate from the many instances of it. One may propose, however, the 2015 Ford Mustang. No tokens of this car exist, but merely mentioning it likely will conjure up such a (mental) thing.

Wetzel: There is, in fact, genetic diversity at the species level. Genetic abnormalities are analogous to mispronunciation of words, and typehood is inhibited in neither case.

3) Interbreeding Natural Populations, Reproductively Isolated (Mayr)

Wetzel: There are hybrids through interspecies breeding. Inter-type substitution of word- or letter-tokens is possible with minimal loss of comprehension.

4) A Lineage (Simpson)

Wetzel: (After Dupre) such a characterization presupposes that those earlier up the line must have been subjected to some other characterization to create the line in the first place. Words' etymologies would inherit similar suppositions.

The point of her exhaustive comparison is that "biologists do pretty well"[16] even though there is nothing that all and only *members* of a particular *species* have in common other than being *members* of that *species*. Substitute "token" for *member* and "type" for *species*, and we're left with Wetzel's original conclusion about word-types. Therefore, since biological taxonomy is successful in its discipline, so is the token/type distinction in linguistics. Types are real kinds (only) inasmuch as species are.

Wetzel's argument culminates in a compact definition of word-types: "abstract theoretical linguistic entities."[17] She acknowledges that a complete definition must also include criteria for distinguishing word types from *other* "abstract theoretical linguistic entities," like sentence-types and letter-types, but this preliminary characterization is of interest here. Her justification of the second term in the characterization, *theoretical*, is most attractive: word-types are "entities that are postulated by, and whose existence is

[16] Wetzel, "On Types," 252.
[17] Ibid., 255.

justified by the success of, an important scientific theory, just as species are."[18] The attractiveness is twofold. For one thing, tokens are not sorted automatically or naturally, nor are they supernaturally preordained as members of certain types. It is a theory (*postulated by* the human mind) doing the sorting. For another, types are not so abstract as to be hypothetical; their *existence is justified* by theory. The types are there, but it is up to us to apprehend them. Successful attempts to do so will be flexible enough to account for exceptions and abnormalities; unsuccessful ones will simply disappear along with the types they intended to grasp.

1.1.3 Pitch-types

Borrowing from Wetzel's definition of word-types, I will attempt to characterize one possible musical type: a pitch.

*A **pitch-type** (pitch) is an abstract music-theoretical entity.*[19]

Let us take the three components of this characterization one at a time.

A pitch-type is abstract, while "frequency" is an attribute of a concrete token. Just as *cat* is a word-type, *High A* is a pitch-type. A cat is not merely an inscription with three parts, the first of which being a circle 75 percent complete. Nor is it merely a monosyllabic utterance pronounced "kæt." These are descriptions of certain concrete cat-tokens. Likewise middle C is abstract, but one of the phenomena perceived at the beginning of a recording of Chopin's *Ballade, Op. 38* is a concrete token. After critically

[18] Ibid., 253-4.
[19] Pitch-type here roughly equates to what we call pitch. Some examples of pitch-types: "High B", A440, G_4, Middle C, +21. Pitch-type is *not* pitch class. Pitch classes are shown in Section 6 to be just that: classes. They are higher-level sortals. If you will, types of types.

reviewing three familiar versions of the abstract/concrete distinction,[20] Bob Hale in *Abstract Objects* proposes a concise account of the distinction:

> "*F* is an abstract sortal iff, for any *R* that grounds *F*, either (i) *R* cannot hold between spatially located items at all, or (ii) *R* can hold between things which are spatially, but not temporally, separated."[21]

In this formulation, *F* = sortal (for our purposes, type) and *R* = equivalence relation. Let us take for example the sortal *pet* and the equivalence relation *furry critter living in my house*. There are two beings that, under this equivalence relation, would be sorted as *pets*. They are spatially but not temporally separated (they exist in different places at the same time), meeting criteria (ii). Therefore *pet* (as defined by my equivalence relation) is abstract. Criterion (ii) is designed so that concrete objects fail. For example the sortal *Fluffy* and the equivalence relation, *feline acquired on July 16, 2007 by this author* would fail because while Fluffy can be temporally but not spatially separated (on the windowsill every morning), he cannot be spatially separated but not temporally separated (in two places at once). Returning to the *Ballade*, we notice that *Middle C* easily passes criterion (ii)—a pianist may sing the "tenor" part while playing. We also notice that *the middle C that begins Chopin's Ballade* also passes—two piano students can begin their renditions simultaneously in adjoining practice rooms. Criterion (i) is designed so that higher-level sortals will still be deemed abstract. A *musical parameter*, for example, is such a sortal.

---

[20] Bob Hale, *Abstract Objects* (New York: Basil Blackwell, 1987), 46-50. Hale discusses problems with three familiar assessments of the distinction: *accessibility to the senses* (that concrete objects are seen, touched, etc.); *acausality* (that abstract objects are acausal); and *spatiality and temporality* (that concrete objects exist in space and time). He also discusses at length what he calls the *ostension* and *functional* criterions in Michael Dummett's perspective, which "roughly is that a concrete object—a particular tree for example—may be singled out ostensively by means of a demonstrative phrase, e.g. 'that elm', accompanied by a pointing gesture: whereas an abstract object—a particular shape or direction say—cannot be so identified ; we have rather to refer to it as the shape of that window, say, or as the direction of such and such a line or movement." Hale has problems with the latter half (the functional criterion) as does Dummett himself, but the ostension criterion is quite rough-and-ready: if you can point at it, literally or figuratively, it is concrete. See *Abstract Objects*, 50.

[21] Ibid., 61.

The equivalence relation that grounds it—say, *an essential attribute of a musical sound*—cannot hold between spatially located items; it passes criterion (i) and is, thus, abstract. It is a higher-level sortal whose equivalence relation holds between the abstract items pitch, duration, amplitude, and timbre. It is a type of types.

A pitch-type is music-theoretical because it is music theory that gives the pitch its identity, its name. Furthermore, music theory, not mere acoustics, determines of what type a given token is. After all, if a piano is completely out of tune, does middle C cease to be middle C? What possible alternate key, when attempting to play the *Ballade*, should one strike instead? Given the constraints of the performer, we must allow the instrument's disagreeable middle C be a token of the type, in order for the performance to continue (and to remain a token of its "*Ballade*-type").[22] Instruments built in different keys—and the music written for them—illustrate similar cases. If the *Ballade* were to be transcribed for, say, French horn choir (never mind *why*), the piano's middle C would be written a G4 for one horn part, while the piano's later F3 would be middle C for another. Both notes are still tokens of the type middle C, but in totally different senses. It is the music-theoretical understanding of score, part, and transposition that help sort the type assignments. Enharmonic spellings are yet another example. Pitches A# and Bb may well be tokens of the same type if either they are simple instructions for someone to produce a single note on an equal-tempered instrument or one is chosen over the other by the composer out of deference to an overwhelmed performer dealing with copious accidentals, but they certainly are not tokens of the same type when they belong to a piece in B minor or are realized in just intonation. Again, music theory becomes the arbiter, or sorter, of token and type. With regard to all of the above, different musicians

[22] See Wolterstorff's "improperly formed examples" in footnote 4.

may disagree or may have different tolerances, but like Wetzel's token-types, there is nothing that pitch-tokens have in common—not frequency, not spelling, not notation—other than being members of the same pitch-type.

Finally, a pitch-type is an entity. Wetzel's word-tokens are "instances of the same type, and this alone is what they all have in common. So the word (type) itself is very important—a very important *entity*. This is evident from the fact that we have names for words."[23] We know pitch-types are important entities because, except in certain computer music, composers do not instruct performers to perform this or that *frequency*. A composer certainly could do so, but to use names of pitch-types is far preferable. Not only are they part of a system and language that performers understand, they have meaning *beyond* frequency. Their relationships with other pitch types, in a scale, say, are wrapped up in their very names. Even pitch-integer notation in twelve-tone equal temperament suggests far more than does the frequency alone. It establishes the smallest measurement in the system (the semitone—perhaps more meaningful than the arbitrary hertz), it lets us then measure intervals in whole numbers, and it facilitates entire theories (set theory, transformational theory, etc.). Such developments (as well as the development of tonal theory) and transmission of any attendant meaning would be inefficient at best, impossible at worst, without the use of pitch-types.

1.1.4 Relationship Between Type and Token

If the concept of type and token is acceptable, one might inquire as to the relationship between the two. Are types defined by the tokens, vice versa, or is it a little of both? To illuminate the relationship between a type and its tokens, Richard Wollheim

[23] Wetzel, "On Types," 253.

contrasts it with two other relationships between a generic entity and its element.[24] One is that found between a *class* and its *members*; the other, between a *universal* and its *instances*.[25] All three relationships are assessed by what he calls their "intimacy."

The least intimate on Wollheim's scale is the class/member relationship. "[A] class is merely made of, or constituted by, its members which are extensionally conjoined to form it."[26] Summing up a class simply requires a dispassionate accounting of all of its members. The class *Current Sprint Employees* is simply the roster of those who currently are employed by Sprint. As employees come and go, the class size changes accordingly. If there were to be an employee meeting and not all employees (members) showed up, those absent would contribute to the incompleteness of the meeting (class). Not so for types and tokens. In fact it would be difficult to imagine a meeting of all tokens of middle C. As fast as they arrive, as many more could be composed, heard or dreamed up. Furthermore, even if the opening of the *Ballade* did not show up, the type Middle-C would be no less incomplete. Similarly arguing that types are not simply classes of their tokens, W.V. Quine eloquently describes *imagined* types, the sentences not yet formed:

> What about two little lines of pentameter that are fated never to get thought up? Taken as classes of their tokens, each of the lines is identically the empty class; so there is but one. This we find unacceptable. We do not want to say that every line of pentameter, save one, is destined some day to be uttered or written.[27]

---

[24] Wollheim's blanket expressions *generic entity* and *element* will be used generically henceforth when we don't want to specify one of his three types of relationships.
[25] Wollheim, *Art*, 75-6.
[26] Ibid., 76.
[27] W.V. Quine, *Quiddities: An Intermittently Philosophical Dictionary* (Cambridge, MA: Belknap, 1987), 217.

Because classes are defined by their members, Quine wonders about those without any members at all. For example, we would have to say that 22nd-century composers are equivalent to 23rd-century ones. As classes they are devoid of members, both the null set, equally empty. Thinking of them as types, however, we might imagine a 22nd-century token composing on the moon and a 23rd-century token composing in another galaxy—clearly, *different* tokens. Some generic entities may be classes *or* types depending on our meaning. There is a *class* of 21st-century composers. If you have composed in this century, you are a member of this class. There is also a *type* 21st-century composer. "A doctor, a lawyer, and a 21st-century composer walk into a bar…" The joke teller describes three types, while the joke listener simply conjures up three individual tokens. With regard to music theory, what may come to mind is the pitch class, and it is indeed a class unlike the pitch-type. Membership in this class is defined simply by octave and enharmonic equivalence.[28] Pitch-class C (pc0) could no more *ex*clude C_5, than it could *in*clude $F\#_2$. Although the names of pitch classes reflect human intervention, they are presumed before the sorting process. Moreover, the name of the class itself is drawn from that of its members. In this way, classes are defined *bottom-up*.

More intimate on Wollheim's scale is the universal/instance relationship. "[A] universal is present in all its instances. Redness is in all red things."[29] A universal is a generic entity, sometimes understood as a property or an attribute. More so than types or classes, universals are rejected by nominalists to various degrees. When Wollheim flatly states that redness is in all red things, he implies that redness must first *be* a thing if it is

[28] Milton Babbitt, "Twelve-Tone Invariants as Compositional Determinants," *Musical Quarterly* 46/2 (1960): 246-59, reprinted in *The Collected Essays of Milton Babbitt*, ed. Stephen Peles, Stephen Dembski, Andrew Mead, Joseph N. Straus (Princeton, NJ: Princeton University Press, 2003), 55-69.
[29] Wollheim, *Art*, 76.

to *be in* something else. These so-called "red things" to a nominalist do not have redness; they simply *are* red. To be red is a predicate, they say, something many objects can *do*. Universals will not be so quickly rejected here. I agree they are different from types—one can not conjure a token of redness, or a token of humor—but I appreciate the common-sense, practical use of an abstract term that is object not predicate. We do use the term redness, after all, even if it is difficult to imagine *a* redness. A sense of humor is not the same as *being funny*. Adopting universals for their ease and practicality admittedly amounts to a considerably uncritical stance, but it is taken with deference in general to music —an abstract art whose commentators rely on universals for a living—and in particular to a useful distinction, one that exemplifies classes and universals: the distinction between pitch class and chroma. Chroma can refer either only to saturation (intensity) of a color, or to combination of saturation and hue, the latter effectively being a synonym for color itself.[30] The term made the leap into music by way of psychology where initial studies of absolute pitch perception referred to a pitch's name without respect to octave displacement as its *chroma* and referred to a pitch's register as its *tone height*.[31] It was rather fortunate that the term pitch class was not to be coined until several decades later and thus unavailable, for *chroma* fittingly has all the hallmarks of a universal, not a class; and when those with absolute pitch perceive a tone, it is said that they immediately apprehend its chroma as if the tone is endowed with a recognizable "color." This is not literally a color but a perceivable attribute. If the perception instead involved comparison to other tones displaced by one or more octaves—what good

---

[30] *Merriam-Webster Online Dictionary*, s.v. "chroma," http://www.merriam-webster.com/dictionary/chroma (accessed June 2008).
[31] See Géza Révész, *Einführung in die Musikpsychologie* (Berne, 1946), translated as *Introduction to the Psychology of Music* (Norman: University of Oklahoma Press, 1953).

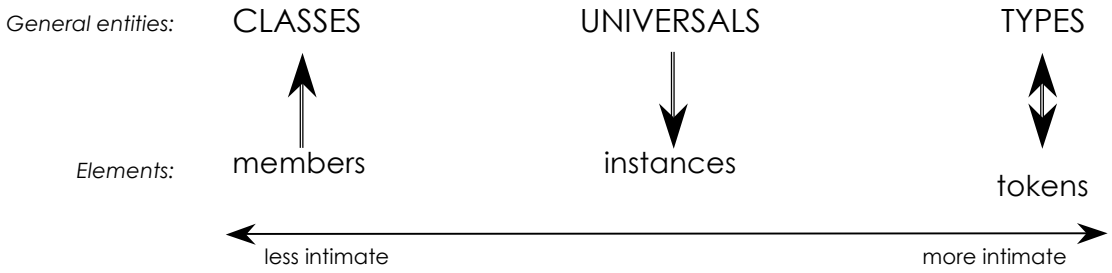
relative pitch skill enables—then perhaps pitch class is what is recognized only after apprehension of its members. Research in this area is far from complete, and it is surely difficult for those without absolute-pitch ability (this author included) to understand fully its nature, but the distinction—chroma as universal, pitch class as class—is useful even at the level of token as we shall see below. For the moment, let us simply contrast the *bottom-up* definition of classes with that of universals. Since universals are understood to inhabit their instances, one may imagine objects in the concrete world simply waiting for attributes to be bestowed from the abstract world. One might imagine objects coming into existence in the concrete world bearing the attributes assigned to them in the abstract one. In either case the definition is *top-down*. The property or attribute itself is abstract and is given to the object. To reverse this direction is to reject outright the universal. A bottom-up perspective simply makes a predicate out of the attribute.

Most intimate on Wollheim's scale is the type/token relationship. "Not merely is the type present in all its tokens like the universal in all its instances, but for much of the time we think and talk of the type as though it were itself a kind of token, though a peculiarly important or pre-eminent one."[32] This is true in talk of pitch-types and pitch-tokens. For some music one may say, "the melody *is* la-la-la." One won't say, "the melody-type *has as one of its possible tokens* la-la-la." Earlier discussion of tokens and types is sufficient to set them apart from the other elements and generic types, but we may benefit from displaying them all together. Figure 1.1 shows Wollheim's three relationships with arrows between generic type and element to show direction of definition. Note that the token/type relationship, the most intimate, includes a two-way arrow. It is not always clear if the definition is top-down or bottom-up. For pitch tokens

[32] Wollheim, *Art*, 76.

and types, it may be either. Asking someone to play A440 can be to define the token by the type, top-down in the manner of a universal—"play note until the needle on the tuner

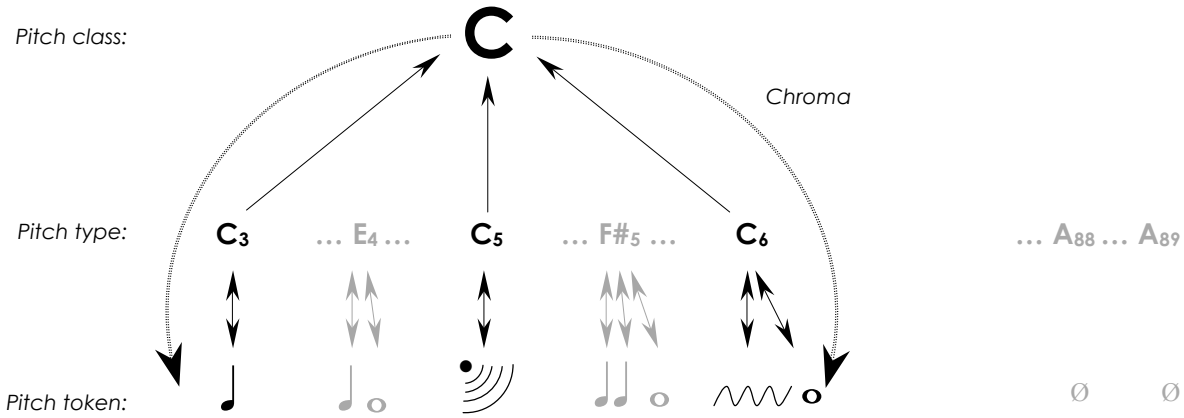
Figure 1.1 Wollheim's general entities and elements on scale of intimacy. Vertical arrows indicate direction of definition.



1.1.5 Pitch and Pitch Class

Pitch tokens, pitch types, pitch classes, and chroma are different entities, ontologically distinct. Figure 1.2 displays these entities as well the relationships between them. The lowest level contains pitch tokens, the actual notated or heard entities. They cluster underneath the pitch types of which they are tokens. Pitch types lead with upward

Figure 1.2 Relationships among pitch tokens, pitch types, pitch classes, and chroma.



Points to A440." It also can be to define the type by the token in the manner of a class—"the tone coming from my tuner is *an* A440. Can you please give me another one?" arrows toward the pitch class they constitute. Dotted arrows emanate downward from the pitch class directly to the pitch tokens. These represent the chroma, which, as a universal, may directly inhabit the concrete elements below. This way, any perceived note can be analyzed either as a token of a type, or an instance of a universal. The pitch types make up a necessary intervening level between the pitch classes and the pitch tokens. It is impossible to propose a token directly of pitch-class C without it. Any physical manifestation of one must be fixed in some register as a pitch type. The only "direct route" from pitch class to token available to us is that from universal (chroma) to instance.[33]

At the far right of the figure there are the (hypothetical) pitch classes A_{88} and A_{89}. Although there exist corresponding frequencies in the magnetic spectrum, there are no musical examples of these types. As argued before, their identical null-set constitution precludes their being classes, so they are here included as types. Their imagined tokens are, of course, not identical; they are an octave apart.

1.1.6 Occurrences

In order to explore types and tokens in pcmultisets, we first need one more related term that is neither token nor type. Occurrence is a term born of the necessity to keep our ontological levels in order.[34] When someone warbles "Happy Birthday to You," four tokens of the word-type *happy* are sung. No problem here. In fact, in earlier literature,

[33] Perhaps a shepherd tone is a case where chroma is less esoteric and more graspable.
[34] Quine's definition of occurrence is refined in Linda Wetzel, "What Are Occurrences," 215-20.

tokens were synonymous with occurrences.[35] But let us consider the song itself as a composition or composition-type. How many times does *happy* appear? Four, of course, but we cannot call these four "happies" tokens, for we are still at the abstract level of types, ontologically prior to instantiation. We can, however, call them occurrences. They are occurrences of the type *happy*. To do so we must "jettison the belief that occurrences are tokens."[36] Occurrences allow types to have repeated segments or portions. When one sits (again) to play the *Ballade*, a series of *tokens* of middle C will be produced by the left hand. The composition, existing outside of notation or performance as an abstract *type*, on the other hand, begins with several *occurrences* of the pitch-type middle C. This distinction will be essential when as compare the formation of pitch-class multisets to that of traditional pitch-class sets.

1.1.7 MST vs. CST

Figure 1.3 shows the formation of pitch sets (psets) and pitch-class sets (pcsets) under classical set theory (CST). The opening measures of Schoenberg's Piano Piece, Opus 19, No. 4 are displayed in the example as pitch tokens (ptokens). At the first level above, there are the pitch types (ptypes), formerly just "pitches" of which all the notated elements are tokens (middle C = 0). The first note is a token of the ptype 17. The second note is a token of the ptype 21. The third note is another token of the ptype 17, but the figure does not include another 17 at the level of ptype because the pset by definition does not allow multiple occurrences of the ptype 17—or of any other ptype. So at the level of ptype there will only be one occurrence of 17, and the resulting pset will contain

[35] Ibid., 215.
[36] Ibid., 216.

Figure 1.3 Formation of psets and pcsets under CST. Schoenberg, Op. 19, No. 4. Cardinality equivalence forces consolidation of multiple ptokens into ptype 17 and multiple ptypes into pc9.

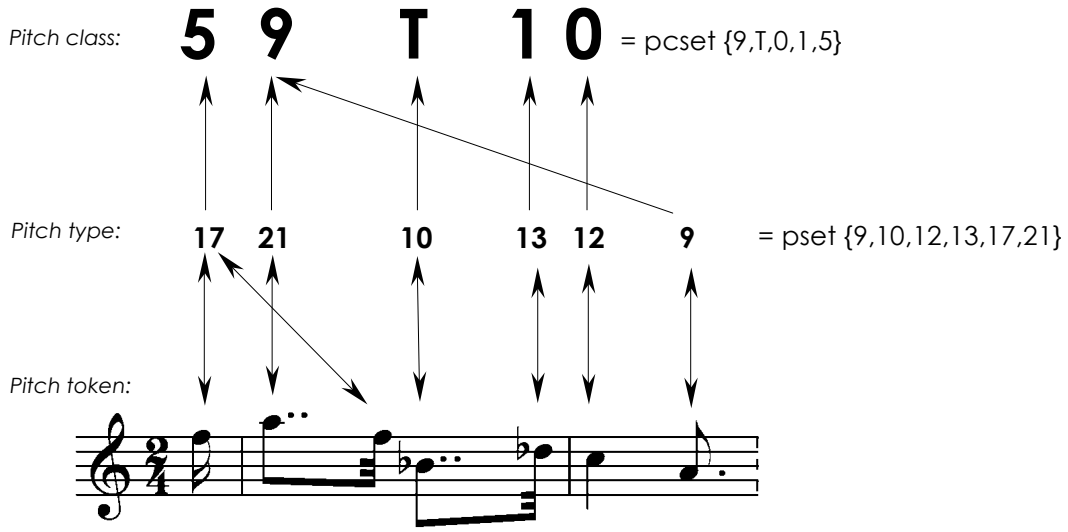
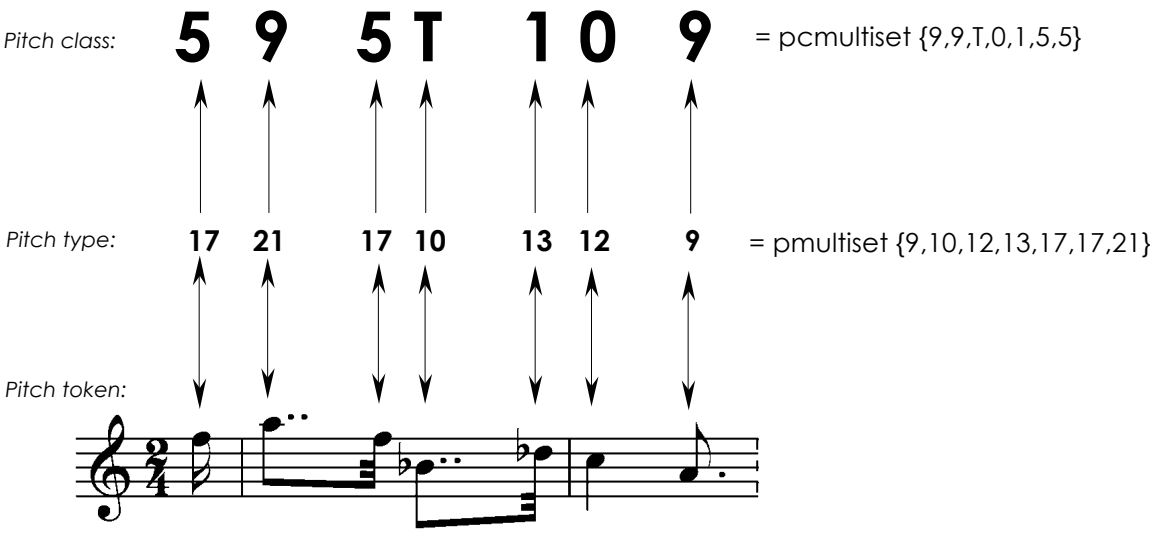


Figure 1.4 Formation of pmultisets and pcmultisets under MST. Schoenberg, Op. 19, No. 4. Removal of cardinality equivalence allows for multiple occurrences of a ptype and multiple occurrences of a pclass.



Single occurrences of ptypes: {9,10,12,13,17,21}. Moving up to the pitch-class level requires a similar consolidation. Ptypes 9 and 21 are both members of the pitch class 9. Since CST does not allow multiple appearances of a pc in a pcset, the arrows indicate consolidation into pc9 so that the resulting pcset contains one appearance of each pc: {9,T,0,1,5}.

Figure 1.4 shows formulation of pmultisets and pcmultisets. *Pitch multisets may contain multiple occurrences of a pitch type.* Therefore, moving up from ptoken level to ptype level requires no consolidation. Two occurrences of ptype 17 are allowed to appear, comprising the pmultiset {9,10,12,13,17,17,21}. Moving up to the pitch-class level, we need to allow a single class to appear twice in order to retain the multiplicity that was evident at the ptoken level. The fundamental difference here between CST and MST (the difference between Figs. 1.3 and 1.4) is this: CST 1) can recognize pitch equivalence between two tokens of the same ptype, but 2) through cardinality equivalence prohibits multiple occurrences of a type in the formation of a set.[37] MST does 1) but *not* 2).

I now modify the notion of occurrences of a type and propose *representatives* of a pitch class. When we list pcs in a pc multiset we are simply listing the class to which each type belongs. In figure 1.4, since there are two occurrences of ptype 17, and each is a member of pc 9, the pcmultiset will list two pc 9s because *pitch-class multisets may contain multiple representatives of a pitch class.* This, of course, is prohibited in CST, and the case for this prohibition is that classes contain *all* of its members. Listing or otherwise displaying a class twice, then, is either meaningless or redundant or both.

[37] See Cliff Callender, Ian Quinn, Dmitri Tymozcko, "Generalized Voice-Leading Spaces," *Science* 320 (April 18, 2008): 346-48 and Dmitri Tymozcko, "The Geometry of Musical Chords," *Science* 313 (July 7, 2006): 72-74. In their geometric spaces, these authors typically disregard cardinality equivalence.

Using the notion of *representatives*, though, allows us in a way, to reflect the chroma or the universal side of pitch classes. Earlier I wrote that a pitch class is strictly a class but also a universal, a chroma. Pitch-class multisets are, in essence, a direct apprehension of the tokens' chromas. For example, when we read the Schoenberg excerpt we might say, "F, A, F, Bb, Db, C, A." When we do so, we are ignoring register (ptype), but we definitely do not mean "All Fs, all As, all Fs, all Bbs, etc." It is understood that there is only one token at a time, and we are listing the chroma of each. So, now moving from pitch type to pitch class, once again CST 1) can recognize octave equivalence between members of a pitch class, but 2) disallows multiple representatives of a pitch class in the formation of a class set. MST does 1), but not 2).

Pitch multisets may contain multiple occurrences of a pitch type, and pitch-class multisets may contain multiple representatives of a pitch class, but the number of elements in either will always equal the number of pitch tokens under study. This now-extended definition is important to remember when dealing with multisets. The pcset {4,7,9} and the pcmultiset {4,7,9} look identical in their notation but are far different in their meaning. CST will not tell us how many representatives of the constituent pcs in pcset {4,7,9} are to found in a given musical scenario. Of course, such an accounting is often irrelevant to the task at hand. When the relative multiplicity of each pitch class *is* relevant, MST is preferred. The pcmultiset {4,7,9} indicates a specific multiplicity of one for each constituent. In this case there is only one representative of each pitch class.

To sum up, I propose two definitions. A **pitch multiset** is an unordered collection of pitch types (not necessarily distinct), in which each element denotes a single occurrence of its pitch-type. A **pitch-class multiset** is an unordered collection of pitch

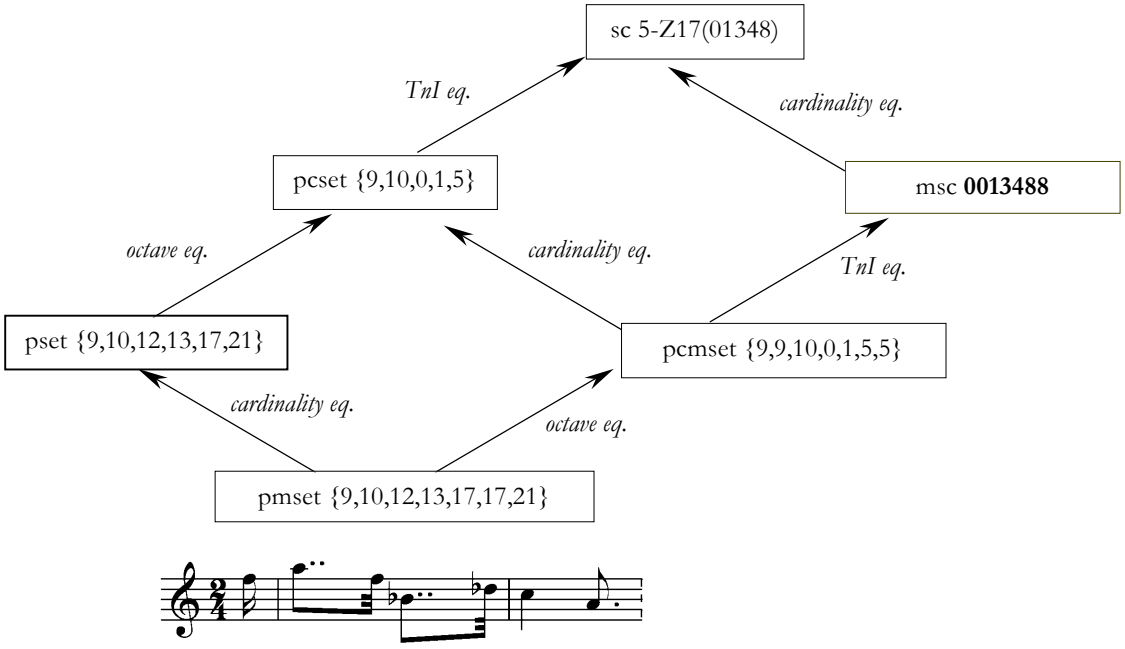
classes (not necessarily distinct), in which each element denotes a single representative of its pitch-class. The multiplicity of pitch-type occurrences and of pitch-class representatives in their respective multisets can, for analytical purposes, model the multiplicity of concrete pitch tokens in a given musical scenario. Frequently it will be a case of repeated pitch classes—as in the Schoenberg above. In some cases it will be doubling among instruments. In others the multiplicity will be only figurative; there may be some reason to give one pitch class a heftier representation than another. As we will see, the tools and machinery already developed in CST are just as useful in MST, where in some cases they offer greater insight into the nature of our musical elements.

1.2 Terminology of Pitch-Class Multisets

A set class in classical set theory (CST) generalizes T_n/T_nI-equivalent pitch-class sets. A multiset class in multiset theory (MST) generalizes T_n/T_nI-equivalent pitch-class multisets. Using the Schoenberg excerpt from Figure 1.3, Figure 1.5 displays its multiset class (msc 0013488) in equivalence relations not only with its set class, but also with the previously discussed set-theoretical apparatuses. CST provides (from specific to general): the pitch set (pset) of pitch types, the pitch-class set (pcset) of pitch classes, and the set class (sc). MST similarly provides: the pitch multiset (pmset) of occurrences of pitch types, the pitch-class multiset (pcmset) of representatives of pitch classes, and the multiset class (msc). This chapter introduces terminology regarding aspects of multiplicity in pcmultisets, considers normal and prime forms of pcmsets, and explores

the relationship at the pinnacle of Figure 1.5—that between set class and multiset class. The application of a *multiplicity function* to a single set class is shown to produce many multiset classes; a *parent* set class has an infinite number of *off-sprung* multiset classes.

Figure 1.5 A collection of notes expressed in six different ways. The pmultiset is "closest" to the score. Others lie at different levels of abstraction due to various types of equivalence.

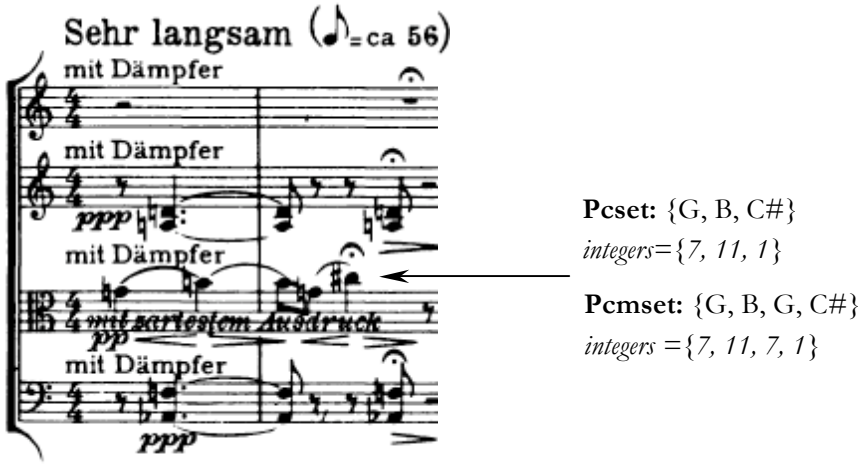


1.2.1 Elements and objects

Figure 1.6 contains the first two measures of Webern's Op. 5, No. 2. As seen in the example, the first four notes (pitch tokens) in the viola part—G, B, G, and C#—can be bundled as the trichordal pcset {7,11,1}, but when comprising a pcmultiset, each of the four notes is represented by an integer—7, 11, 7 (again), and 1. There are four pitch-class representatives in the pcmultiset, all separated by commas, each one standing in for the pitch class of a single pitch token in the music. These four representatives are the

four elements of the pcmultiset. The discrete pitch classes being represented, 7, 11, and 1, are the three objects of the pcmultiset. The pitch classes are, basically, objects to which the elements point. They are the target of the elements' reference. Therefore, the

Figure 1.6 Webern, Five Movements for String Quartet, II, Op. 5, No. 2, mm. 1-2.



Pcmultiset {7,11,7,1} (or {7,7,11,1}—more on normal form below) has four elements and three objects. There can be more elements than there are objects, but there cannot be more objects than there are elements. In other words, more than one element may represent the same pitch class, but a single element may not represent more than one pitch class.

Cardinality of a pcset is defined by the number of its elements. We similarly define cardinality of a pcmultiset, and thus the following two pcmultisets are both deemed tetrachords: {1,1,1,6} and {2,3,4,4}. It would be helpful, though, to affix to the cardinality (tetra-) a modifier indicating the number of objects in the multiset. Therefore, we label {1,1,1,6} a **2-object tetrachord** and {2,3,4,4} a **3-object tetrachord**. This

nomenclature allows the reader to identify easily the number of both elements and objects in pcmultisets of varying size and constitution.[38]

It is important to remember that a pcmultiset may have the same number of objects and elements and still be a pcmultiset. Pcmultiset {2,3,4} and pcset {2,3,4}, for instance, may be notated similarly, but the pcmultiset's pc representatives refer to *exactly* three pitch tokens, while the pcset's pcs refer to *at least* three.

1.2.2 Multiplicity

In a pcmultiset, the number of representatives of a certain pitch class (or the number of elements pointing to a certain object) is the **multiplicity** of that pitch class (or object).[39] Webern's pcmultiset {7,11,7,1}, found in Figure 1.6, contains pc7 at a multiplicity of two. Robert Morris suggested in *Composition with Pitch-Classes* that this multiplicity can be represented with superscripts.[40] Such power notation appears as early as 1961 in Milton Babbitt's "Set Structure as Compositional Determinant," representing multiplicity not of equivalent pcs, but of equivalent partitions of the aggregate.[41] Thus, {7,11,7,1} would become $\{7^2,11,1\}$. This compression is helpful when the number of elements far exceeds the number of objects or pitch classes (i.e., when there is massive doubling), but when the pcmultisets are of manageable size, it is easier simply to have the doubled (or tripled) object appear as two (or three) independent full-scale integers.

---

[38] The converse is also possible but much less preferred. For example, one might label the pcmultiset {1,1,1,6} a *4-element dyad*, possibly noticing the similarity to the dyad {1,6}. This reluctance to consider it a tetrachord, defining cardinality by number of objects, undermines the very purpose of the pcmultiset. Each of the three pc1 representatives is a member (element) of the pcmset, each as full-fledged as the pc6 representative. The pcmultiset is useful *because* it allows a collection like {1,1,1,6} to be a tetrachord.
[39] Likewise, in a pmultiset, the number of occurences of a certain pitch type is the multiplicity of that pitch.
[40] Robert B. Morris, *Composition With Pitch-Classes* (New Haven: Yale University Press, 1983), 333n.
[41] Milton Babbitt, "Set Structure as a Compositional Determinant," *Journal of Music Theory* 5/1 (1961): 72-94, reprinted in *The Collected Essays of Milton Babbitt*, ed. Stephen Peles, Stephen Dembski, Andrew Mead, Joseph N. Straus, (Princeton, NJ: Princeton University Press, 2003), 86-108.

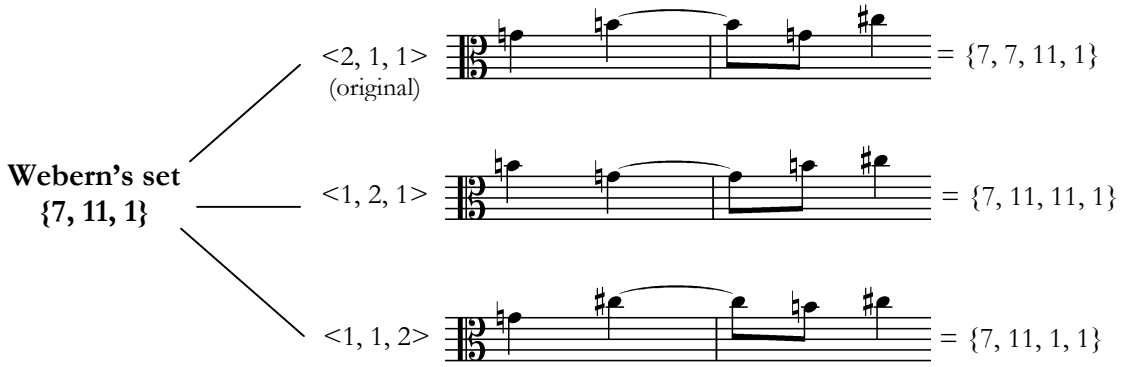
One efficient way to display multiplicity in a pcmltiset, while concealing actual pcs, is to use a **multiplicity function**. The function is defined as <a, b, c…> for any pcmultiset $\{x^a, y^b, z^c \ldots\}$. Each position in the function represents the multiplicity of the corresponding element in the pcmultiset. (For now, a, b, c, … will be positive integers, but in a later section I consider zeros and negative integers: negative membership.) The mfunction helps us to produce the various possible pcmultisets derived from doubling of various pcs in a pcset. For example, the viola's melody in Figure 6 is a statement of the pcset {7,11,1}, a trichord. But one object, pc7, is doubled on the musical surface, producing the three-object tetrachord {7,7,11,1}. Put another way, the multiplicity function <2,1,1> is *applied* to the pcset {7,11,1} yielding the pcmultiset {7,7,11,1}. A multiplicity function will have as many positions as objects (pcs), and the sum of the entries will equal the number of elements. Therefore, a three-object tetrachord has the following three possible multiplicity functions: <2,1,1>, <1,2,1>, and <1,1,2>. Since each of the three entries must be at least 1 (by definition there are three objects), and since the sum of the entries must be four (there are four elements in a tetrachord), these are the only three possibilities.[42]

Showing what nuances pcmultisets have to offer, Figure 1.7 recomposes Webern's melody using the same pitch-classes but different multiplicity. On the first line is Webern's four-note melody extracted from the previous example. Displayed this way, the melody results from the application of the multiplicity function <2, 1, 1> to the pcset {7, 11, 1}. Hypothetical melodies resulting from the other two possible vectors are given below the original. All three melodies undeniably are musical realizations of the same

[42] To find the multiplicity vectors of a pcmultiset with x objects and y elements, find the number of compositions of y with x terms. More in section 1.3.

pcset, but each is a unique pcmultiset. Of course, the different resulting contours contribute to the different melodic characters, but the repeated pc in each case also has significant influence regardless of contour. Since the B and the C# are closest to each

Figure 1.7 Different multiplicity functions applied to Webern's pcset {7,11,1}.

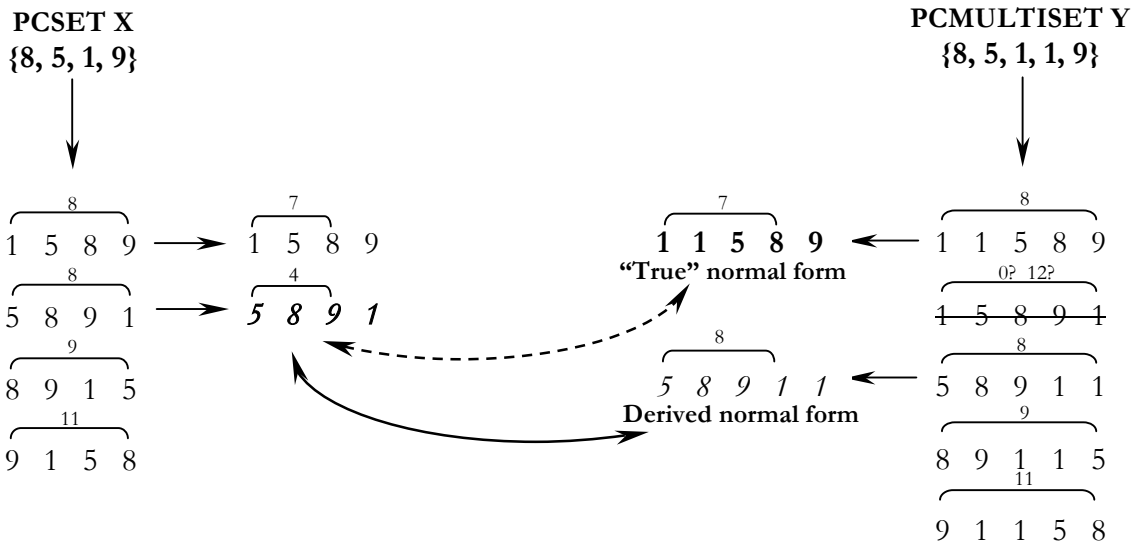


Other (in pc space or when the set is as compact as possible in p space) it is by doubling the G that Webern creates the sense of reaching further—first four semitones (G-B), then six (G-C#). The second line in the example (mfunction: <1, 2, 1>) seems more balanced around B, while the third line (mfunction: <1, 2, 1>) has a focus on C# where the B is a close neighbor and the G is remote memory. Pitch-class sets always have been understood to be generalizing tools. As a collection of integers, the pcset tells nothing about dynamics, durations, repetitions, or duplication, nor should it. This example demonstrates how pcmultisets can address the latter two aspects by taking the pcset as a starting point and further refining it, formally and precisely addressing characteristics previously unanalyzed or left to analytical prose. Even relative dynamics and duration can be tackled with multisets.

1.2.3 Normal Form

To write effectively a pcmultiset in normal form is not as simple as one might expect, and our method ultimately presents two possibilities. Let us first follow the standard procedure for pcsets, i.e. to rank the pcs in ascending order within an octave, to compare all rotations, and choose the one with the smallest span from first pc to last. Ties are broken in the usual fashion: compare spans from first pc to penultimate, then, if necessary from first to antepenultimate, and so forth. Figure 1.8 displays this procedure performed upon both the pcset X {8, 5, 1, 9} and the pcmultiset Y {8, 5, 1, 1, 9}; neither

Figure 1.8 Determining normal forms of a pcset and a similar pcmultiset by standard procedure.



Is in normal form to start. It should be clear that multiset Y is related to pcset X through the application of a multiplicity function. All four rotations of pcset X are shown on the left, and two of them have the same span of 8 semitones. This tie is broken by a comparison of spans from first to penultimate pcs. The rotation {5, 8, 9, 1} proves to be the best candidate for normal form having a tiebreaking span of 4 semitones.

Pcmultiset Y, on the right side of the example, allows five rotations, one of which poses a minor difficulty. The rotation {1, 5, 8, 9, 1} (struck-through in the example) has a span of either 0 semitones or 12 semitones. A span of 0 semitones is unacceptable because it simply disregards the pitch-class space consumed by the other members. A span of 12 might intuitively make more sense. Indeed, the span begins at pc1, passes through some other pcs, then returns to pc1. This trip spans 12 semitones, but such a span breaks one of the rules of normal form construction. Pcs must be ranked in ascending order *within* an octave. For pcsets, which have no doubled pcs, the maximum possible span of any such ranking following this rule would be eleven semitones. It is never a concern whether "within" means "less than" or "less than or equal to" because the span never could be exactly an octave. Now, pcmultisets present the possibility of an octave span. We must now take "within" definitively to mean "less than." This way, any rotation of a pcmultiset that spans an octave, thus separating one of the doubled pcs, is not a candidate for normal form.

Of the remaining eligible rotations there is a tie between {1, 1, 5, 8, 9} and {5, 8, 9, 1, 1}. Each spans eight semitones. Because of the doubled pc1, the latter rotation still spans eight semitones when the penultimate pc is considered in a tiebreaker. Rotation {1, 1, 5, 8, 9}, with a penultimate span of seven semitones is thus deemed the "true" normal form. Having two pc1s at the beginning, this normal form certainly is "packed to the left," but it has a distinct disadvantage. Its formerly clear relationship to the pcset is now disguised. Pcmultiset Y {8, 5, 1, 1, 9} clearly can be derived from pcset {8, 5, 1, 9} simply by doubling pc1. If this relationship is to remain evident in the normal forms, one might prefer the rotation {5, 8, 9, 1, 1} as the normal form of the pcmultiset. Its ordering

of pcs remains unchanged. This option is indicated in the example with a solid line and labeled "derived." This normal form is actually easier to produce. One simply finds the normal form according to standard procedure, *with doublings omitted*, as one would for the traditional pcset. In other words, one finds normal form as always, but does so with *objects* rather than *elements*. Once normal form is found, the doublings are simply reinstated. Either normal form, derived or true, may be preferable depending on analytical context, but when we consider prime forms (when we establish multiset *classes*), it will be even more important to maintain the clear relationship between pcset and related pcmultiset. We will want the ordering in the pcmultiset to "match" that in the pcset regardless of doubling. Therefore when we use "normal form" it is in the "derived" sense.

1.2.4 Multiset Classes

Just as a single set class represents a collection or equivalence class of T_n/T_nI-related pcsets, so does a **multiset class** represent families of pcmultisets. As discussed above, the ordering of pcs in a pcmultiset's normal form, despite the various multiplicities of each member, is to be the same as that in a traditional pcset containing the same pcs at multiplicities of one. In the same way, the ordering of distinct integers in a multiset class reflects one of the traditional set classes. For example, the pcset {7, 11, 1} is a member of set class 3-8 (026). That is, pcset {0,2,6} is the representative pcset of the class. Since original pcset and set-class representative are related by T_1I, the following mapped pairs are manifest: 7 onto 6, 11 onto 2, and 1 onto 0. Any pcmultiset that contains the original three (and only these three) objects—7, 11, and 1—will belong

to a multiset class containing the integers 0, 2, and 6. Furthermore, the multiplicity of occurrences of pcs in the pcmultiset will be matched by the corresponding representatives in the multiset class. So, as pcset {7,11,1} belongs to set class 3-8 (026), pcmset {7,7,11,1} belongs to multiset class 0266.

A multiset class labeled 0266 might incite two opposing intuitions. First, negatively, this ordering of integers is "packed to the right," that is, the interval between final and penultimate pc representatives is smaller than ever before. It is a unison. One might be inclined instead to invert this configuration, producing 0046. While this result does have the appealing double zero at the front end, and it may be more left-packed, it runs counter to the second intuition. No matter how many 0s, 4s, or 6s may appear, those three integers do not appear together in any of the canonical trichord set classes. Intuitively, 0266, despite its right-heaviness, seems to resonate with our concept of a "026 trichord." After all, it contains the same integers. One easily can interpret the pcmultiset 0266 as a modified pcset 026. It is as if the "6" simply cloned itself right in place. Indulging this intuition, we see that each traditional set class has an infinite number of related multiset classes. Set class 026, for example, is related to multiset class 0266 but also to multiset classes 00266, 00026, and even 002226666. To find, then, the multiset class to which any pcmultiset belongs, we find prime form in the same fashion as we found normal form: we arrange by object, not by element. One first reduces multiplicity of each pc to one, then determines the set class in the usual fashion, and finally reinstates the multiplicity of each corresponding object. When encountering pcmultiset {7,7,7,11,1}, for example, one should trace the mapping of {7,11,1} to its set

class, 026, and reinstate the multiplicity of pc7 at the representative, 6, resulting in multiset class 02666.

In the previous example we discovered multiset class 0266. By choosing a label intuitively more appealing than the reasonable 0046, we established a connection between multiset class and set class. The familiar set class 026 has new kin: 0266, 0226, 0026, etc. Here, 026 is the **parent** set class, and the multitude of others are the **off-sprung** multisets. A single parent can have an infinite number of offspring because the potential multiplicity of any object is infinite, but it is important to remember that a parent set class also can produce a single multiset class whose elements are each at a multiplicity of one. As shown above, the pcmultiset {7,11,1} does exist distinct from the pcset {7,11,1}. The former implies *only one* of each pc, while the latter implies *any number* of each pc. Similarly, the multiset class 026 is distinct from the set class 026. The off-sprung multiset classes result from the multiplication, or intensification, of one or more (or no) objects, while the parent set classes determines which objects will be put into play. Therefore, to maintain clarity, we will refer to set classes with Forte notation—3-8 (026)—and the multiset classes with integers alone—026, 0026, 0226, 0266, etc. It is fitting to display the generic parent class with its almost arbitrary, hyphenated appendage while displaying the off-sprung multiset class's element members unencumbered by parentheses. We are thus reminded that each discrete integer represents at some level a single discrete pitch token.

1.3 Mapping the terrain

The multiplicity function (mfunction) is a convenient tool for examining multiplicity in pcmultisets of various cardinalities. As shown in the previous section, it displays multiplicity of objects in a pcmultiset—it counts representatives of each pc—without reference to specific pitch classes. When all possible mfunctions are applied systematically to a parent class (one of the canonical set classes) we may discover all of its possible off-sprung pcmultisets up to a given cardinality.

In this section we map this terrain. First we apply all possible multiplicity functions (up to twelve elements total) to each of the 222 set classes from monad to dodecachord. Noting that T-symmetrical and I-symmetrical set classes produce fewer unique off-sprung multiset classes, we then carefully reduce for T_n/T_nI equivalence. In short, we find 113,973 unique multiset classes with 12 or fewer elements.

1.3.1 Mfunctions as compositions

In the previous section we demonstrated how a pcset can generate a pcmultiset through the application of a multiplicity function. The trichord {3,5,9} generates the 3-object tetrachords {3,3,5,9}, {3,5,5,9}, and {3,5,9,9} through the application of the vectors <2,1,1>, <1,2,1>, and <1,1,2> respectively. These three mfunctions—the only such functions for generating tetrachords from trichords (without zeros or negative numbers)—can be seen, in mathematical terms, as different *compositions* of the number four. A composition is a representation of a positive integer that lists possible positive integers of which it is a sum. Unlike in a partition, different orderings of the integers are considered different compositions. For example, the 32 compositions of the number 6

include the following: 6, 5+1, 1+5, 4+2, 2+4, 4+1+1, 1+4+1, 1+1+4, etc. Notice that 1+5 and 5+1 are distinct compositions. The total number of compositions of any given number *x* is simply 2^{x-1}.

We can refine the number of compositions by choosing how many terms (how many integers) the desired compositions should contain. For example, the five compositions of 6 into only two terms are 5+1, 1+5, 4+2, 2+4, and 3+3. Calculating the total number of compositions of a given number with a given number of terms, uses the binomial coefficient, or "choose" function as follows:

$$\text{Compositions of } x \text{ with } y \text{ terms} = \binom{x-1}{y-1} = \frac{(x-1)!}{y-1!((x-1)-(y-1))!}$$

Calculating the number of possible mfunctions for a specific number of elements and objects uses the same formula, defining *x* = *number of elements* and *y* = number of objects. This gives us the following:

$$\text{mfunctions with } x \text{ elements and } y \text{ objects} = \frac{(x-1)!}{y-1!((x-1)-(y-1))!}$$

This formula is indispensible when finding the mfunctions for higher numbers of elements.

Figure 1.9 displays the total number of mfunctions whose objects and elements number twelve or fewer.[43] The shaded 1s represent the mfunctions whose objects and elements are equal in number (i.e., <1>, <1,1>, <1,1,1>, etc.). Columns display total

[43] For this project, I've limited myself to 12 elements. This is somewhat arbitrary, but capping the enumeration at twelve insures that the largest pcmsets considered are no larger than the aggregate itself.

Figure 1.9 Total number of mfunctions for objects and elements numbering up to twelve.

	1 ob.	*2 ob.*	*3 ob.*	*4 ob.*	*5 ob.*	*6 ob.*	*7 ob.*	*8 ob.*	*9 ob.*	*10ob.*	*11 ob.*	*12 ob.*	*Totals*	
1 elem.	**1**												**1**	*2^0*
2 elem.	1	**1**											**2**	*2^1*
3 elem.	1	2	**1**										**4**	*2^2*
4 elem.	1	3	3	**1**									**8**	*2^3*
5 elem.	1	4	6	4	**1**								**16**	*2^4*
6 elem.	1	5	10	10	5	**1**							**32**	*2^5*
7 elem.	1	6	15	20	15	6	**1**						**64**	*2^6*
8 elem.	1	7	21	35	35	21	7	**1**					**128**	*2^7*
9 elem.	1	8	28	56	70	56	28	8	**1**				**256**	*2^8*
10 elem.	1	9	36	84	126	126	84	36	9	**1**			**512**	*2^9*
11 elem.	1	10	45	120	210	252	210	120	45	10	**1**		**1024**	*2^10*
12 elem.	1	11	55	165	330	462	462	330	165	55	11	**1**	**2048**	*2^11*
Totals	**12**	**66**	**220**	**495**	**792**	**924**	**792**	**495**	**220**	**66**	**12**	**1**		

Number of mfunctions with a given *number of objects* (i.e., number of different pcs, number of places in the mfunction, or number of terms in the composition). Rows display a total number of mfunctions with a given *number of elements* (i.e., actual pc representatives, the sum of mfunction entries, the number being composed into terms).[44] Total number of mfunctions for each number of objects is shown across the bottom, while total number of mfunctions for each number of elements is listed down the right side. The latter total (the total number of compositions into *any* number of terms) is also calculated down the far right side with the formula 2^{x-1}.

To show what the numbers in the table represent, we will produce the septachord multiset classes off-sprung from parent set class 4-1(0124). (These would be the multiset classes with seven elements but only four pcs: 0, 1, 2, and 4.) To find the total number of mfunctions that can produce a 4-object septachord, we look in Figure 1.9 where "*4-ob.*"

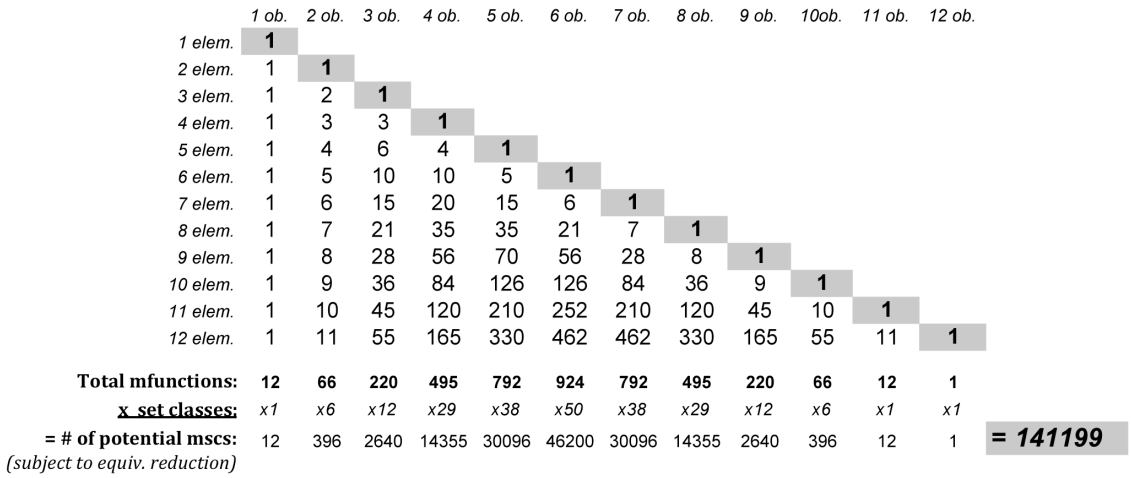
[44] Because the binomial coefficient is used in this calculation, the results arrange themselves in the fashion of Pascal's Triangle.

Figure 1.10 Twenty possible mfunctions with 4 objects and 7 elements. Each is applied to set class 4-2(0124), yielding twenty multiset classes.

MFUNCTION	MSETCLASS
<4,1,1,1>	0000124
<3,2,1,1>	0001124
<3,1,2,1>	0001224
<3,1,1,2>	0001244
<2,3,1,1>	0011124
<2,2,2,1>	0011224
<2,2,1,2>	0011244
<2,1,3,1>	0012224
<2,1,2,2>	0012244
<2,1,1,3>	0012444
<1,4,1,1>	0111124
<1,3,2,1>	0111224
<1,3,1,2>	0111244
<1,2,3,1>	0112224
<1,2,2,2>	0112244
<1,2,1,3>	0112444
<1,1,4,1>	0122224
<1,1,3,2>	0122244
<1,1,2,3>	0122444
<1,1,1,4>	0124444

And "*7-elem.*" Meet, and we find 20; there are twenty mfunctions meeting these criteria. Figure 1.10 systematically lists these twenty mfunctions, starting with the highest first entry and continuing from there. Each mfunction is applied to SC 4-2(0124), and the twenty off-sprung multiset classes are given to the right of the mfunction. No two of these multiset classes are equivalent through transposition or inversion because the parent set class, 4-2(0124) is neither T- nor I-symmetrical. We might choose to repeat this process, applying the same twenty mfunctions to each of the 29 canonical tetrachord set classes, producing 580 potential off-sprung multiset classes. They are merely "potential" because an accounting must be made for the multitude of duplications due to the T- and I-symmetry of the parent set classes. We do so below, but first let us continue the process and apply our all mfunctions to all of the canonical set classes. Figure 1.11 reproduces

Figure 1.11 Mfunctions applied to all of the canonical set classes.



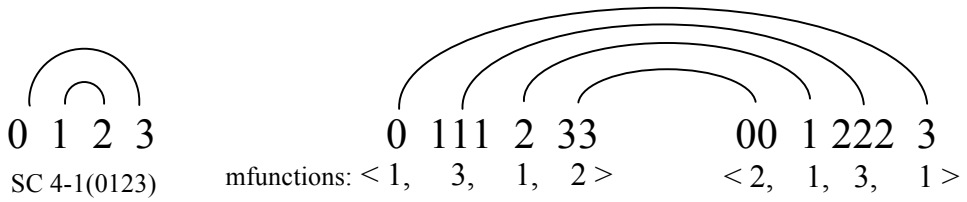
	1 ob.	2 ob.	3 ob.	4 ob.	5 ob.	6 ob.	7 ob.	8 ob.	9 ob.	10ob.	11 ob.	12 ob.	
1 elem.	**1**												
2 elem.	1	**1**											
3 elem.	1	2	**1**										
4 elem.	1	3	3	**1**									
5 elem.	1	4	6	4	**1**								
6 elem.	1	5	10	10	5	**1**							
7 elem.	1	6	15	20	15	6	**1**						
8 elem.	1	7	21	35	35	21	7	**1**					
9 elem.	1	8	28	56	70	56	28	8	**1**				
10 elem.	1	9	36	84	126	126	84	36	9	**1**			
11 elem.	1	10	45	120	210	252	210	120	45	10	**1**		
12 elem.	1	11	55	165	330	462	462	330	165	55	11	**1**	
Total mfunctions:	**12**	**66**	**220**	**495**	**792**	**924**	**792**	**495**	**220**	**66**	**12**	**1**	
x set classes:	*x1*	*x6*	*x12*	*x29*	*x38*	*x50*	*x38*	*x29*	*x12*	*x6*	*x1*	*x1*	
= # of potential mscs: *(subject to equiv. reduction)*	12	396	2640	14355	30096	46200	30096	14355	2640	396	12	1	***= 141199***

Figure 1.9, but the results across the bottom—the total number of mfunctions (up to 12 elements) for each number of objects—are now multiplied by the number of canonical set classes of each cardinality. Each product is the total number of potential multiset classes (with 12 elements or fewer) for any number of objects (any parent class), *still subject to T_n/T_nI equivalence*. For example, 495 mfunctions can be generated with four objects and up to twelve elements. (There are 495 compositions of 4, 5, 6,… 12 into 4 terms.) If each of the 495 mfunctions is applied to each of the 29 canonical tetrachords, we find 14,355 potential 4-object multiset classes. When this is repeated for every canonical set class of all cardinalities, we find 141,199 potential multiset classes. Our next, more difficult, task is to reduce this number, accounting for all of the T- and I- equivalent results stemming from symmetrical parent set classes.

1.3.2 Multiplicity and Symmetry

Figure 1.12 shows the inversionally symmetrical transformational mappings inherent in 4-1(0123). Next to this, one potential multiset class, 0111233, is created by applying a single mfunction <1,3,1,2>. Then, another potential multiset class 0012223 is

Figure 1.12 Unique mfunctions applied to a symmetrical parent, yielding equivalent results.



Similarly created using a different mfunction <2,1,3,1>. As their inversional mappings show, these potential multiset classes are, in fact, the same multiset class, thanks to their T_nI equivalence. At each end of a given mapping we find same number of pc representatives: two 0s map onto two 3s, three 2s map onto three 1s, and so forth. Such equivalence is possible, because of the corresponding multiplicities of each member, but it would be impossible had the parent set class not been inversionally symmetrical in the first place. In other words, without the mappings provided by the single parent set class, we would not have equivalence among multiple off-sprung multiset classes. In essence, then, to find symmetry in a msetclass we look in its mfunction for mappings that match precisely the mappings in the parent class. Figure 1.13 lists the twenty possible 4-ob./7-elem. Mfunctions and applies each to two different parent set classes: 4-2(0124) and 4-1(0123). Set class 4-2(0124) produces twenty unique multiset classes with seven elements while 4-1(0123) produces only ten. Because 4-1(0123) has at least one degree

Figure 1.13 The 7-elem./4-obj. mfunctions applied to two different parent set classes

MFUNCTION	4-2(0124)	4-1(0123)	
<4,1,1,1>	0000124	0000123	
<3,2,1,1>	0001124	0001123	
<3,1,2,1>	0001224	0001223	
<3,1,1,2>	0001244	0001233	
<2,3,1,1>	0011124	0011123	
<2,2,2,1>	0011224	0011223	
<2,2,1,2>	0011244	0011233	
<2,1,3,1>	0012224	0012223	
<2,1,2,2>	0012244	*(0012233*	*= 0011233)*
<2,1,1,3>	0012444	*(0012333*	*= 0001233)*
<1,4,1,1>	0111124	0111123	
<1,3,2,1>	0111224	0111223	
<1,3,1,2>	0111244	*(0111233*	*= 0012223)*
<1,2,3,1>	0112224	*(0112223*	*= 0111223)*
<1,2,2,2>	0112244	*(0112233*	*= 0011223)*
<1,2,1,3>	0112444	*(0112333*	*= 0001223)*
<1,1,4,1>	0122224	*(0122223*	*= 0111123)*
<1,1,3,2>	0122244	*(0122233*	*= 0011123)*
<1,1,2,3>	0122444	*(0122333*	*= 0001123)*
<1,1,1,4>	0124444	*(0123333*	*= 0000123)*
	20 mscs	**10 mscs**	

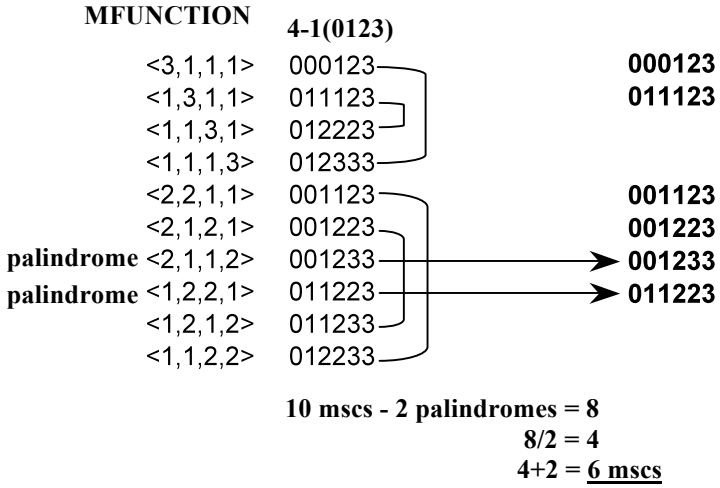
of inversional symmetry, some applications of the mfunctions will be inversionally equivalent to others. For this reason, the results contain twenty *unique* msetclasses (one for each of mfunction) when applied to the asymmetrical 4-2(0124) but half as many when applied to 4-1(0123), which, itself, contains symmetry. Each mfunction has a retrograde or mirror image, an inversionally symmetrical partner. There are 27 other tetrachord parent classes, but exploring each one in the same detail is not necessary because we can predict that similar application of the twenty mfunctions will produce 20 msetclasses if the parent is asymmetrical and 10 msetclasses if the parent shows the same symmetry as does 4-1(0123). At first glance, one might assume that we simply divide the

total number of potential msetclasses by two whenever inversional symmetry appears. This assumption, however, ignores two points essential to correct enumeration: 1) Some mfunctions are inversionally symmetrical *with themselves*. 2) There are different *types* of inversional symmetry. A third point will aid the process: 3) In an mfunction, what is relevant with regard to symmetry is not the actual multiplicity in each position of the function, but the number of appearances of each distinct multiplicity. (Mfunctions <1,2,2,1> and <4,3,3,4> are equivalent with regard to symmetry—note the placement of like multiplicities. More generally, the partitions 2+2+1+1 and 4+4+2+2 can produce the same number of symmetrical mfunctions because they both have exactly *two* of some pc, and exactly *two* of some other.) An examination of these three points follows below.

Palindromic Compositions

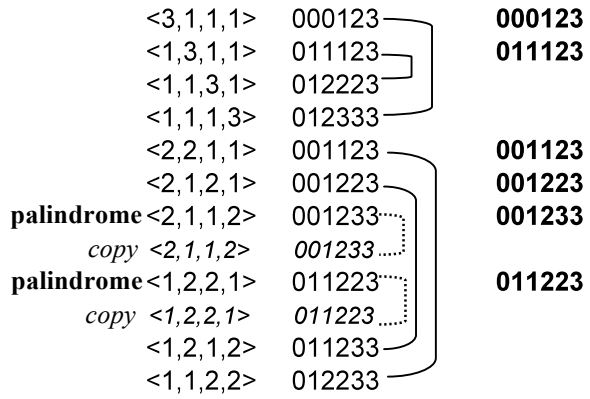
Each of the 7-elem./4-obj. mfunctions listed in Figure 1.13 has an inversional partner; none of these mfunctions is inversionally symmetrical with itself. That is, none of the compositions (mfunctions) is a palindrome (retrograde of itself). Other mfunctions, though, depending of the number of its elements and objects, will be palindromic compositions. Figure 1.14a lists all of the 6-elem./4-obj. mfunctions (the 10 compositions of 6 into 4 terms). Two of these mfunctions, <2,1,1,2> and <1,2,2,1>, are palindromic compositions, each its own symmetrical partner. The other eight mfunctions are symmetrically paired off with partners. To find the number of msetclasses by simply dividing the number mfunctions in half would, therefore, be incorrect. The two inversionally symmetrical mfunctions must first be withheld before the total is halved. Withhold two from ten (= 8), halve (= 4), and replace the two withheld (= 6). Figure 1.14a lists the final six unique msetclasses to the right. It will be to our advantage,

Figure 1.14a The 6-elem./4-obj. mfunctions, applied to parent set class 4-1(0123). (Method 1)



10 mscs - 2 palindromes = 8
8/2 = 4
4+2 = 6 mscs

Figure 1.14b The 6-elem./4-obj. mfunctions, applied to parent set class 4-1(0123). (Method 2)



10 mscs + 2 palindrome copies = 12
12/2 = 6 mscs

$$\frac{\#T_0\text{-sym.}(10) + \#\text{I-sym }(2)}{\text{total degrees of sym. of parent }(2)} = \#\text{ of mscs}$$

however, especially when we approach multiple degrees of symmetry, not to withhold the self-symmetrical mfunctions before halving and replacing but instead to add a copy of each before halving. Figure 1.14b demonstrates how this procedure gives the same results. (Compare Figures 1.14a and 1.14b.) Not only does this procedure save us the trouble of "withholding" and "replacing," it also allows us simply to add the number of mfunctions symmetrical at T_0 (all ten in this case) to the number of I-symmetrical mfunctions (the two palindromes) before dividing by two (the total degrees of symmetry).[45] More on this generalization to come.

Types of Inversional Symmetry

The canonical set classes exhibit, to varying degrees, transpositional (T-) symmetry, inversional (I-) symmetry, or both. The *degrees* of either type of symmetry indicate the number of specific operations (of that type) that will map the members of the set class onto themselves (and/or each other) in a one-to-one fashion.[46] One degree of T-symmetry refers always to the operation T_0. Every set class is symmetrical by at least this degree (T_0), but higher degrees of T-symmetry, though rare compared to those of I-symmetry, are simple to assess and pose few problems in mfunctions. Two degrees of T-symmetry always refer to the operations T_0 and T_6. Three degrees, to T_0, T_4, and T_8 and so forth; the degrees of symmetry divide the set or set class into as many partitions, each

[45] At first glance, the identification of palindromic mfunctions might seem problematic for a parent class such as 3-9(027). Applying the mfunction <1,1,3> gives the inversionally symmetrical 02777, but <1,1,3> is apparently not palindromic; <1,3,1> is. If one rotates all compositions of this partition <3,1,1>, <1,3,1>, and <1,1,3> by one position, however, one should easily find not only the same three compositions, but also the same number of symmetrical ones. Similarly, one could simply consider a rotation of the set class, 270, instead of the prime form, 027. Order is important, but the specific rotation is not. Display on a clock face would make this even clearer. There is one palindromic mfunction in this case, and it can be rotated as needed, so long as all others are similarly rotated to suit the set class.

[46] See Joseph N. Straus, *Introduction to Post-Tonal Theory*, 3rd ed. (Upper Saddle River, NJ: Pearson Prentice Hall, 2005): 82-91, 261-64.

segment a transposition of the others. I-symmetry is a bit more complex. When a set class possesses one degree of I-symmetry, it is not immediately clear which inversion—more specifically, which *index* of inversion—is the one that produces the required self-mappings. It is equally unclear at first whether the inversional axis runs *through* a pitch class, mapping it onto itself, or *between* two pitch classes, mapping them onto each other. These matters must be considered before anticipating how many multiset classes spring off from a given parent. What is more, if an axis does, in fact, run though two pitch classes, we must determine whether the set class actually contains members at one, both, or neither of those positions.

The index of inversion represents an axis, around which pitch classes in a pcset (or set class) may symmetrically map onto one another, but it more specifically refers to a sum, the sum of any symmetrically mapped pair of pitch classes.[47] When the sum is 0, 2, 4, …10, the index is "even," and when the sum is 1,3,5,…11, the index is "odd." Even indices allow pitch classes to map onto themselves. (Any whole number, when added to itself, becomes an even number.) Odd indices disallow such self mapping. Axes associated with even indices run *through* pitch classes, while those associated with odd indices run *between* them. Using a partional approach, Brian Alegant has sorted the set classes into EVEN and ODD collections, and all of the I-symmetrical set classes fall into one or the other or both "inventories".[48] We will borrow his EVEN and ODD terminology here, but a further distinction within the EVEN family must be made. We will do so using the following labels:

---

[47] See, again, Milton Babbitt, "Twelve-Tone Invariants."
[48] Brian Alegant, "When Even Becomes Odd: A Partitional Approach to Inversion," *Journal of Music Theory* 43/2 (Autumn 1999): 193-230.

EV^0 – Ascribed to those EVEN set classes whose axis of inversion runs through two pitch-class positions, neither of which belongs to the set class. Example: SC(0134) – the axis runs through pc2 (and pc8), but neither belongs to this set class. Inversion entails the mapping of two discrete pairs, and there are *no* self-mappings. Only set classes of even cardinality can have this symmetry.

EV^1 – Given to those EVEN set classes whose axis of inversion runs through two pitch-class positions, exactly one of which belongs to the set class. Example: SC(01234) – the axis runs through pc2 (and pc8), but only one of them belongs to this set class. Inversion entails the mapping of two discrete pairs, and there is *one* self-mapping. Only set classes of odd cardinality can have this symmetry.

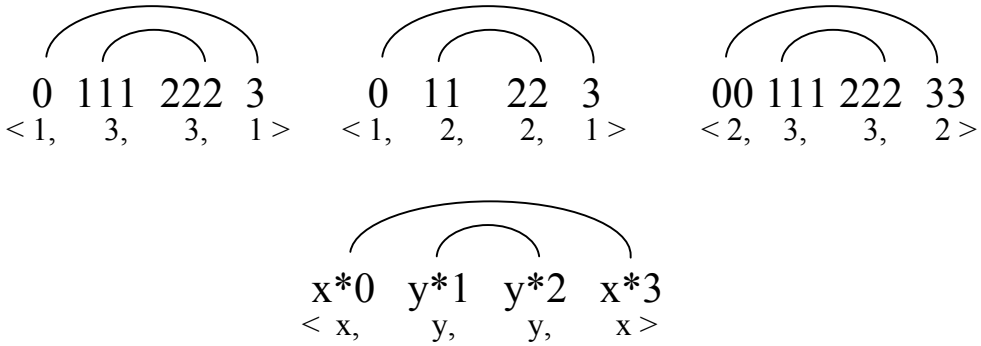
EV^2 – Given to those EVEN set classes whose axis of inversion runs through two pitch-class positions, both of which belong to the set class. Example: SC(012348) – the axis runs through pc2 (and pc8), and both of them belong to this set class. Inversion entails the mapping of two discrete pairs, and there are *two* self-mappings. Only set classes of even cardinality can have this symmetry.

Set classes in Alegant's ODD inventory—those whose axes run between pitch classes—need no further distinction and simply are labeled ODD. In fact, we shall find that when applying the mfunction to set classes, ODD symmetry in a parent class produces the same results as does EV^0; they both contain exclusively pairwise mappings with no self mappings.

Multiplicity of Multiplicity

When the number of elements or objects is small, it is quite simple to list those mfunctions that are symmetrical to the same degree that the parent is. As our multiset

Figure 1.15 Functionally equivalent mfunctions (with respect to symmetry) generalized to compositions of multiplicity.



Classes increase in number of objects and elements, the need for a more systematic approach becomes evident. Therefore, we might label the mfunctions according to the multiplicity of the terms in its composition. The terms in an mfunction represent, in turn, multiplicity of pcrepresentatives, so we actually must look at multiplicity of multiplicity! Figure 1.15 brings back the mfunction <1, 3, 3, 1> and compares its application to SC 4-1(0123) to applications of mfunctions <1, 2, 2, 1> and <2, 3, 3, 2>. The three mfunctions produce different multiset classes, but each has the same symmetrical mappings because SC 4-1(0123) has the mappings "permanently wired," and we need only confirm that the multiplicities are analogous. Below these in the figure is the same mapping, each multiplicity in the mfunction is now replaced by the variables x and y. This generalizes many mfunctions into a single one, each variable representing *some* multiplicity. (In two cases, y=3; in another, y=2.) What matters is *how many* of each variable there are. This way, if we know the effect of the symmetrical mfunction <x,y,y,x>, we need not measure the effects of <1, 2, 2, 1>, <1, 3, 3, 1>, <2, 1, 1, 2>, etc. Furthermore, if the partition 2+2+1+1 has two palindromic compositions (mfunctions

<2,1,1,2> and <1,2,2,1>), so does the partition 3+3+1+1, and even 6+6+5+5. All three partitions have a *multiplicity of multiplicity* (MM) of 2-2. That is, there are two appearances of one multiplicity and two of another. The well-known formula provided in the previous section gives the number of compositions of a given number into a given number of terms, but when we know further *which* terms we prefer, the formula is simpler: the total number of objects (factorial) is divided by the product of the multiplicity of each term (factorial). Figure 1.16 shows the six compositions of a tetrachord with MM 2-2 (two pcs at one multiplicity, two at another). Where previously we found the total number of mfunctions for a particular number of objects and elements, we now find the number of mfunctions for a given multiplicity of multiplicity regardless of the actual number of elements. Partitions 2+2+1+1, 3+3+1+1, 5+5+3+3, and 4+4+2+2 all would produce the same number of unique mfunctions (6) because they all display the same MM: 2-2.

Figure 1.16 Finding number of compositions of given number into terms, each with specific multiplicity. (Multiplicity of multiplicity (MM) = 2-2)

2 of one term,
2 of another

$$\frac{4!}{2! * 2!} = 6 \text{ compositions:}$$

< x, x, y, y >
< x, y, x, y >
< x, y, y, x >
< y, x, x, y >
< y, x, y, x >
< y, y, x, x >

1.3.3 Enumeration of the Multiset Classes

We are now prepared to enumerate the multiset classes generated by the parent set classes of all cardinalities. Along the way, we will reduce for equivalence resulting from

the symmetry of varying degrees and types found in the parent set classes. Our principal formula is known as Burnside's Lemma. Julian Hook profitably used the formula first to enumerate the 224 canonical set classes, which theretofore had been simply counted up, and second to calculate the number of set classes of a given cardinality.[49] This formula can also be of service to the enumeration of mfunctions given certain symmetry of a parent set class. When a finite group G acts on set X, the number of orbits is equal to:

$$\frac{1}{|G|}\sum_{g\in G}\psi(g)$$

For our purposes, |G| represents the total degrees of symmetry of a given parent class, each individual degree represented by g. The *psi* (ψ) of each g represents the number of mfunctions that are self-symmetrical at that degree of symmetry. In essence, we sum the self-symmetrical mfunctions at each degree of symmetry and divide by the number of degrees; it is an average of self-symmetry. As an example, let us determine the number of 6-element (hexachord) multiset classes produced by sc 4-1(0123). First, we know there are ten 6-elem./4-obj. mfunctions; there are ten compositions of 6 into 4 terms. The sc displays 1,1 EV^0 symmetry, so using Burnside's Lemma we have the following:

$$\frac{1}{2}\left(\psi(T_0)+\psi(I)\right)$$

$$=\frac{1}{2}\left(10+2\right)$$

$$=6$$

[49] See Julian Hook, "Why Are There Twenty-Nine Tetrachords?," *Music Theory Online* 13.4 (December 2007). Hook is able to utilize the binomial coefficient when looking at given cardinalities because in his enumeration each of the twelve pcs is either "on" or "off" with a value of 0 or 1. The number of pcs that have a value of 1 total to the cardinality. Because with pcmultisets, the pcs can be "off," "on," or "on twice," here I would need trinomial coefficents to enumerate only a maximum of doubled multiplicity, a method beyond the scope of this study.

The $\psi(T_0)$ equals the number of mfunctions self-symmetrical under T_0. All ten are, of course. The $\psi(I)$ equals the number of mfunctions equivalent under EV^0 symmetry, i.e. the palindromic compositions. There are two. This formula appears at the bottom of Figure 1.14b, which also displays the six resulting unique mfunctions.

In the scenario above, we recognized that under T_0 all mfunctions are self symmetrical, but to find the two inversionally self-symmetrical mfunctions, first we listed them all (in Figure 1.14b), then we identified the palindromes—a method of brute force. It turns out that some recent research in combinatorics has provided a formula for determining the number of palindromic compositions of a given number into a given number of terms. The authors' theorem is as follows:[50]

The number $P_{n,m}$ of palindromic compositions of n with m summands satisfies:

$$P_{n,m} = \begin{cases} \binom{k-1}{s-1} & \text{if } n = 2k,\ m = 2s \\ \binom{k-1}{s} & \text{if } n = 2k,\ m = 2s+1 \\ \binom{k}{s} & \text{if } n = 2k+1,\ m = 2s+1 \\ 0 & \text{if } n = 2k+1,\ m = 2s \end{cases}$$

For our purposes, n = the number of elements, and m = the number of objects. There are four possible "choose" functions shown. Which to use depends both on whether the number of elements is even (n = 2k) or odd (n = 2k+1) and on whether the number of objects is even (m = 2s) or odd (m = 2s+1). Now, when using Burnside's Lemma, we'll know precisely what number to insert for each degree of inversional symmetry. While in Figure 1.14b it was quite easy to spot the two palindromic mfunctions, this formula not

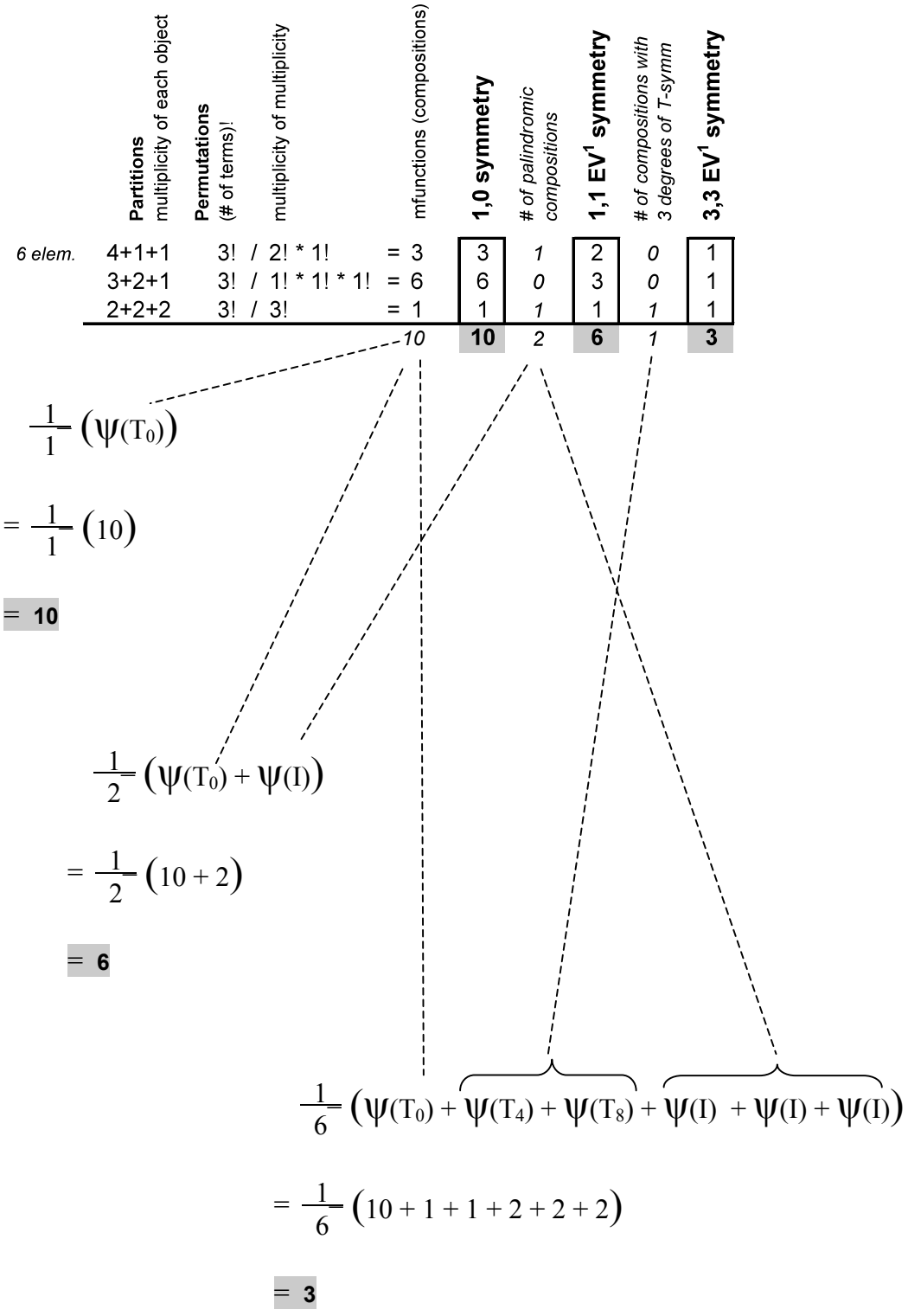
[50] Donatella Merlini, Francesca Uncini, and M. Cecilia Verri, "A Unified Approach to the Study of General and Palindromic Compositions," *Integers: Electronic Journal of Number Theory* 4 (2004), #A23.

only ensures accuracy, but also more elegantly provides results as the total number of mfunctions gets more unwieldy. It is important to note that we will consider palindromic compositions (mfunctions) to be useful in ODD, EV^0, and EV^1, symmetries, but not EV^2 symmetries. In the latter case, the axis runs *through* two pcs in the set class and therefore, inversional symmetry can occur even if the mfunction is not palindromic. For example, the sc (0248), which displays EV^2 symmetry produces an inversionally self-symmetrical offspring using the mfunction <1,2,1,3>: 0224888. The singleton pcs 0 and 4 map to one another while the doubleton and tripleton each maps to itself. The mfunction, however, is not palindromic (in any rotation). Only if 2 or 3 is artificially repeated could we consider a palindrome. More on EV^2 symmetry below.

The Trichordal Parent Classes

We are now prepared to determine the number of 3-object multiset classes for any given number of elements. (We can determine the number of multiset offspring of the 12 trichordal set classes.) Let us first, as a means of demonstration, examine the 6-element/3-object multiset classes (the hexachord offspring of the canonical trichords). Figure 1.17 lists the three partitions of 6 with three terms, and it calculates the number of compostions (mfunctions) for each partition. The three different multiplicity-of-multiplicity (MM) values (2-1, 1-1-1, and 3) can be seen in this calculation. These values are helpful because, as we will see, no matter how many elements we choose, since there are only three objects these are the only possible multiplicity of multiplicities: two pcs of the same multiplicity and one of another; three pcs of different multiplicity; and 3 pcs all of the same multiplicity. So, the results in this small figure of six elements simply can be repeated with any number of elements.

Figure 1.17 Calculating totals of the 6-element offspring of the canonical trichords with 1,0; 1,1 EV^1; and 3,3 EV^1 symmetries



The first boxed column in the figure gives the total number of 6-element multiset classes off-sprung from any trichord with 1,0 symmetry. There are ten. This is equal to the number of possible mfunctions because the asymmetrical (except at T_0) parent will produce one unique offspring for every mfunction. As shown below the figure, Burnside's lemma trivially would provide this answer. The next boxed column gives the total number of 6-element multiset classes off-sprung from any trichord with 1,1 symmetry. This total, 6, is shown below the figure to be a result of Burnside's Lemma. As described above, the *psi* for the single degree of inversional symmetry equals the total number of palindromic compositions. This number, 2, is found in the column between the boxed columns, its total produced in the formula below the figure. The third boxed column in the figure gives the total number of 6-element multiset classes off-sprung from any trichord with 3,3 symmetry. (There is, of course, only one: sc 3-12(048)). Here, Burnside's Lemma also produces the correct value, but now there are six "*psi's*" in the formula. The first *psi* represents the number of mfunctions self-symmetrical at T_0 (all 10). The second *psi* represents the number of mfunctions producing multiset classes self-symmetrical at T_4. (There is only one. This result is found by examining the MM values. If each term in an MM value is divisible by three, then some of the corresponding mfunctions may be self-symmetrical at T_4 and at T_8. Since these T-symmetries require equivalence between each partitioned third of the mfunction, each multiplicity must be divisible by three. For example, each term in the MM value 6-3 is divisible by three some of its mfunctions are T_4-symmetrical. Mfunction <2,1,1,2,1,1,2,1,1>. To determine exactly how many are symmetrical in this case, we simply look at the number of compositions of 2+1+1 (MM = 2-1), "one third" of the original 2+2+2+1+1+1+1+1+1

(MM=6-3). To calculate the number of compositions, then, we do not divide 9! / 6! * 3!. We divide 3! / 2! * 1!. The resulting three are the mfunctions <2,1,1,2,1,1,2,1,1>, <1,2,1,1,2,1,1,2,1> and <1,1,2,1,1,2,1,1,2>. Returning to our trichords, we see one MM divisible by three: 3. We divide by three and determine the number of compositions *not* by dividing 3! / 3! But by dividing 1! / 1!. The result, 1, is given in the column labeled "# of compositions with three degrees of T-sym.") The third *psi* represents the number of mfunctions producing multiset classes self-symmetrical at T_8. As explained parenthetically above, this number is given in its own column. The fourth, fifth, and sixth "*psi's*" all have the value of 2. This represents the number of palindromic mfunctions. The number is found in the column labeled "# of palindromic compositions," its calculation shown below. These six "*psi's*" are summed and then divided by six; there are six total degrees of symmetry in 3,3 symmetry. We now know there are ten hexachordal offspring of each of the following trichordal parents: 3-2(013), 3-3(014), 3-4(015), 3-5(016), 3-7(025), 3-8(026), and 3-11(037); there are six hexachordal offspring of each the following trichordal parents: 3-1(012), 3-6(024), 3-9(027), and 3-10(036); and there are three hexachordal offspring of each of of the trichordal parent 3-12(048).

Figure 1.18 is an expansion of Figure 1.17 to include elements numbering 1 through 12. Totals for each number of elements are shaded in grey. A scan down the multiplicity of multiplicity column will demonstrate what was described earlier: that there are only three possible MM values for trichord parent classes. Grand totals are given at the bottom. If we reproduce the grey shaded totals as well as the grand totals, once for each trichordal parent to which they apply, we would be left with Figure 1.19. This figure charts the total number of 3-object multiset classes with any number of elements

Figure 1.18 Enumeration of the 3-object multiset classes. Symmetries: 1,0; 1,1 EV^1; and 3,3 EV^1

	Partitions multiplicity of each object	**Permutations** (# of terms)!	multiplicity of multiplicity	mfunctions (compositions)	**1,0 symmetry**	*# of palindromic compositions*	**1,1 EV^1 symmetry**	*# of compositions with 3 degrees of T-symm*	**3,3 EV^1 symmetry**
3 elem.	1+1+1	3! / 3!		= 1	1	*1*	1	*1*	1
					1	*1*	**1**	*1*	**1**
4 elem.	2+1+1	3! / 2! * 1!		= 3	3	*1*	2	*0*	1
					3	*1*	**2**	*0*	**1**
5 elem.	3+1+1	3! / 2! * 1!		= 3	3	*1*	2	*0*	1
	2+2+1	3! / 2! * 1!		= 3	3	*1*	2	*0*	1
					6	*2*	**4**	*0*	**2**
6 elem.	4+1+1	3! / 2! * 1!		= 3	3	*1*	2	*0*	1
	3+2+1	3! / 1! * 1! * 1!		= 6	6	*0*	3	*0*	1
	2+2+2	3! / 3!		= 1	1	*1*	1	*1*	1
					10	*2*	**6**	*1*	**3**
7 elem.	5+1+1	3! / 2! * 1!		= 3	3	*1*	2	*0*	1
	4+2+1	3! / 1! * 1! * 1!		= 6	6	*0*	3	*0*	1
	3+3+1	3! / 2! * 1!		= 3	3	*1*	2	*0*	1
	3+2+2	3! / 2! * 1!		= 3	3	*1*	2	*0*	1
					15	*3*	**9**	*0*	**4**
8 elem.	6+1+1	3! / 2! * 1!		= 3	3	*1*	2	*0*	1
	5+2+1	3! / 1! * 1! * 1!		= 6	6	*0*	3	*0*	1
	4+3+1	3! / 1! * 1! * 1!		= 6	6	*0*	3	*0*	1
	4+2+2	3! / 2! * 1!		= 3	3	*1*	2	*0*	1
	3+3+2	3! / 2! * 1!		= 3	3	*1*	2	*0*	1
					21	*3*	**12**	*0*	**5**
9 elem.	7+1+1	3! / 2! * 1!		= 3	3	*1*	2	*0*	1
	6+2+1	3! / 1! * 1! * 1!		= 6	6	*0*	3	*0*	1
	5+3+1	3! / 1! * 1! * 1!		= 6	6	*0*	3	*0*	1
	5+2+2	3! / 2! * 1!		= 3	3	*1*	2	*0*	1
	4+4+1	3! / 2! * 1!		= 3	3	*1*	2	*0*	1
	4+3+2	3! / 1! * 1! * 1!		= 6	6	*0*	3	*0*	1
	3+3+3	3! / 3!		= 1	1	*1*	1	*1*	1
					28	*4*	**16**	*1*	**7**

Figure 1.18 (continued)

	Partitions multiplicity of each object	**Permutations** (# of terms)!	multiplicity of multiplicity		mfunctions (compositions)	**1,0 symmetry**	*# of palindromic compositions*	**1,1 EV[1] symmetry**	*# of compositions with 3 degrees of T-symm*	**3,3 EV[1] symmetry**
10 elem.	8+1+1	3!	/ 2! * 1!	= 3		3	*1*	2	*0*	1
	7+2+1	3!	/ 1! * 1! * 1!	= 6		6	*0*	3	*0*	1
	6+3+1	3!	/ 1! * 1! * 1!	= 6		6	*0*	3	*0*	1
	6+2+2	3!	/ 2! * 1!	= 3		3	*1*	2	*0*	1
	5+4+1	3!	/ 1! * 1! * 1!	= 6		6	*0*	3	*0*	1
	5+3+2	3!	/ 1! * 1! * 1!	= 6		6	*0*	3	*0*	1
	4+4+2	3!	/ 2! * 1!	= 3		3	*1*	2	*0*	1
	4+3+3	3!	/ 2! * 1!	= 3		3	*1*	2	*0*	1
						36	*4*	**20**	*0*	**8**
11 elem.	9+1+1	3!	/ 2! * 1!	= 3		3	*1*	2	*0*	1
	8+2+1	3!	/ 1! * 1! * 1!	= 6		6	*0*	3	*0*	1
	7+3+1	3!	/ 1! * 1! * 1!	= 6		6	*0*	3	*0*	1
	7+2+2	3!	/ 2! * 1!	= 3		3	*1*	2	*0*	1
	6+4+1	3!	/ 1! * 1! * 1!	= 6		6	*0*	3	*0*	1
	6+3+2	3!	/ 1! * 1! * 1!	= 6		6	*0*	3	*0*	1
	5+5+1	3!	/ 2! * 1!	= 3		3	*1*	2	*0*	1
	5+4+2	3!	/ 1! * 1! * 1!	= 6		6	*0*	3	*0*	1
	5+3+3	3!	/ 2! * 1!	= 3		3	*1*	2	*0*	1
	4+4+3	3!	/ 2! * 1!	= 3		3	*1*	2	*0*	1
						45	*5*	**25**	*0*	**10**
12 elem.	10+1+1	3!	/ 2! * 1!	= 3		3	*1*	2	*0*	1
	9+2+1	3!	/ 1! * 1! * 1!	= 6		6	*0*	3	*0*	1
	8+3+1	3!	/ 1! * 1! * 1!	= 6		6	*0*	3	*0*	1
	8+2+2	3!	/ 2! * 1!	= 3		3	*1*	2	*0*	1
	7+4+1	3!	/ 1! * 1! * 1!	= 6		6	*0*	3	*0*	1
	7+3+2	3!	/ 1! * 1! * 1!	= 6		6	*0*	3	*0*	1
	6+5+1	3!	/ 1! * 1! * 1!	= 6		6	*0*	3	*0*	1
	6+4+2	3!	/ 1! * 1! * 1!	= 6		6	*0*	3	*0*	1
	6+3+3	3!	/ 2! * 1!	= 3		3	*1*	2	*0*	1
	5+5+2	3!	/ 2! * 1!	= 3		3	*1*	2	*0*	1
	5+4+3	3!	/ 1! * 1! * 1!	= 6		6	*0*	3	*0*	1
	4+4+4	3!	/ 3!	= 1		1	*1*	1	*1*	1
						55	*5*	**30**	*1*	**12**
			Totals:			**220**		**125**		**53**

Figure 1.19 Multiset Class Totals: Trichordal Parent Classes

	3-1(012)	3-2(013)	3-3(014)	3-4(015)	3-5(016)	3-6(024)	3-7(025)	3-8(026)	3-9(027)	3-10(036)	3-11(037)	3-12(048)	
1 elem.	x	x	x	x	x	x	x	x	x	x	x	x	
2 elem.	x	x	x	x	x	x	x	x	x	x	x	x	
3 elem.	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**12**
4 elem.	2	3	3	3	3	2	3	3	2	2	3	1	**30**
5 elem.	4	6	6	6	6	4	6	6	4	4	6	2	**60**
6 elem.	6	10	10	10	10	6	10	10	6	6	10	3	**97**
7 elem.	9	15	15	15	15	9	15	15	9	9	15	4	**145**
8 elem.	12	21	21	21	21	12	21	21	12	12	21	5	**200**
9 elem.	16	28	28	28	28	16	28	28	16	16	28	7	**267**
10 elem.	20	36	36	36	36	20	36	36	20	20	36	8	**340**
11 elem.	25	45	45	45	45	25	45	45	25	25	45	10	**425**
12 elem.	30	55	55	55	55	30	55	55	30	30	55	12	**517**
	125	220	220	220	220	**125**	220	220	**125**	**125**	220	**53**	***2093***
symmetry	**1,1**	1,0	1,0	1,0	1,0	**1,1**	1,0	1,0	**1,1**	**1,1**	1,0	**3,3**	
inversion	EV^1					EV^1			EV^1	EV^1		EV^1	

up to twelve. Totals for each parent class are given at the bottom, and totals for each number of elements (cardinality) are given at the right. The grey-shaded bar highlights the 3-object trichords, which resemble the twelve canonical trichord set classes.

The Pentachordal Parent Classes

Enumeration of the multset classes with pentachord parent classes is straightforward and similar to that undertaken above; there is only one type of inversional symmetry involved: 1,1 EV^1. Figure 1.20 shows displays the calculation of 792 multiset classes for each parent with 1,0 symmetry and of 416 multiset classes for each parent with 1,1 EV^1 symmetry. Burnside's lemma can be used to confirm each subtotal in the column labeled 1,1 EV^1: add the number of mfunctions (T_0 sym.) to the number of palindromic mfunctions (I sym.) and divide by 2 (1,1 = 2 degrees). Figure 1.21 lists each

Figure 1.20 Enumeration of the 5-object multiset classes. Symmetries: 1,0 and 1,1 EV^1

	Partitions multiplicity of each object	**Permutations** (# of terms)! / multiplicity of multiplicity	mfunctions (compositions)	**1,0 symmetry**	*# of palindromic compositions*	**1,1 EV^1 symmetry**
5 elem.	1+1+1+1+1	5! / 5!	= 1	1	*1*	1
				1	*1*	**1**
6 elem.	2+1+1+1+1	5! / 4! * 1!	= 5	5	*1*	3
				5	*1*	**3**
7 elem.	3+1+1+1+1	5! / 4! * 1!	= 5	5	*1*	3
	2+2+1+1+1	5! / 3! * 2!	= 10	10	*2*	6
				15	*3*	**9**
8 elem.	4+1+1+1+1	5! / 4! * 1!	= 5	5	*1*	3
	3+2+1+1+1	5! / 3! * 1! * 1!	= 20	20	*0*	10
	2+2+2+1+1	5! / 3! * 2!	= 10	10	*2*	6
				35	*3*	**19**
9 elem.	5+1+1+1+1	5! / 4! * 1!	= 5	5	*1*	3
	4+2+1+1+1	5! / 3! * 1! * 1!	= 20	20	*0*	10
	3+3+1+1+1	5! / 3! * 2!	= 10	10	*2*	6
	3+2+2+1+1	5! / 2! * 2! * 1!	= 30	30	*2*	16
	2+2+2+2+1	5! / 4! * 1!	= 5	5	*1*	3
				70	*6*	**38**
10 elem	6+1+1+1+1	5! / 4! * 1!	= 5	5	*1*	3
	5+2+1+1+1	5! / 3! * 1! * 1!	= 20	20	*0*	10
	4+3+1+1+1	5! / 3! * 1! * 1!	= 20	20	*0*	10
	4+2+2+1+1	5! / 2! * 2! * 1!	= 30	30	*2*	16
	3+3+2+1+1	5! / 2! * 2! * 1!	= 30	30	*2*	16
	3+2+2+2+1	5! / 3! * 1! * 1!	= 20	20	*0*	10
	2+2+2+2+2	5! / 5!	= 1	1	*1*	1
				126	*6*	**66**

Figure 1.20 (continued)

	Partitions multiplicity of each object	**Permutations** (# of terms)!	multiplicity of multiplicity	mfunctions (compositions)	**1,0 symmetry**	*# of palindromic compositions*	**1,1 EV[1] symmetry**
11 elem.	7+1+1+1+1	5! /	4! * 1!	= 5	5	*1*	3
	6+2+1+1+1	5! /	3! * 1! * 1!	= 20	20	*0*	10
	5+3+1+1+1	5! /	3! * 1! * 1!	= 20	20	*0*	10
	5+2+2+1+1	5! /	2! * 2! * 1!	= 30	30	*2*	16
	4+4+1+1+1	5! /	3! * 2!	= 10	10	*2*	6
	4+3+2+1+1	5! /	2! * 1! * 1! * 1!	= 60	60	*0*	30
	4+2+2+2+1	5! /	3! * 1! * 1!	= 20	20	*0*	10
	3+3+3+1+1	5! /	3! * 2!	= 10	10	*2*	6
	3+3+2+2+1	5! /	2! * 2! * 1!	= 30	30	*2*	16
	3+2+2+2+2	5! /	4! * 1!	= 5	5	*1*	3
					210	*10*	**110**
12 elem.	8+1+1+1+1	5! /	4! * 1!	= 5	5	*1*	3
	7+2+1+1+1	5! /	3! * 1! * 1!	= 20	20	*0*	10
	6+3+1+1+1	5! /	3! * 1! * 1!	= 20	20	*0*	10
	6+2+2+1+1	5! /	2! * 2! * 1!	= 30	30	*2*	16
	5+4+1+1+1	5! /	3! * 1! * 1!	= 20	20	*0*	10
	5+3+2+1+1	5! /	2! * 1! * 1! * 1!	= 60	60	*0*	30
	5+2+2+2+1	5! /	3! * 1! * 1!	= 20	20	*0*	10
	4+4+2+1+1	5! /	2! * 2! * 1!	= 30	30	*2*	16
	4+3+3+1+1	5! /	2! * 2! * 1!	= 30	30	*2*	16
	4+3+2+2+1	5! /	2! * 1! * 1! * 1!	= 60	60	*0*	30
	4+2+2+2+2	5! /	4! * 1!	= 5	5	*1*	3
	3+3+3+2+1	5! /	3! * 1! * 1!	= 20	20	*0*	10
	3+3+2+2+2	5! /	3! * 2!	= 10	10	*2*	6
					330	*10*	**170**
			Totals:		**792**		**416**

pentachord parent and gives the total number multiset classes for that parent at each number of elements.

Figure 1.21 Multiset Class Totals: Pentachordal Parent Classes.

	5-1(01234)	**5-2(01235)**	**5-4(01236)**	**5-5(01237)**	**5-3(01245)**	**5-9(01246)**	**5-Z36(01247)**	**5-13(01248)**	**5-6(01256)**	**5-14(01257)**	**5-Z38(01258)**	**5-7(01267)**	**5-15(01268)**	**5-10(01346)**	**5-16(01347)**	**5-Z17(01348)**	**5-Z12(01356)**	**5-24(01357)**	**5-27(01358)**
1 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
2 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
3 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
4 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
5 elem.	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
6 elem.	3	5	5	5	5	5	5	5	5	5	5	5	3	5	5	3	3	5	5
7 elem.	9	15	15	15	15	15	15	15	15	15	15	15	9	15	15	9	9	15	15
8 elem.	19	35	35	35	35	35	35	35	35	35	35	35	19	35	35	19	19	35	35
9 elem.	38	70	70	70	70	70	70	70	70	70	70	70	38	70	70	38	38	70	70
10 elem.	66	126	126	126	126	126	126	126	126	126	126	126	66	126	126	66	66	126	126
11 elem.	110	210	210	210	210	210	210	210	210	210	210	210	110	210	210	110	110	210	210
12 elem.	170	330	330	330	330	330	330	330	330	330	330	330	170	330	330	170	170	330	330
	416	792	792	792	792	792	792	792	792	792	792	792	**416**	792	792	**416**	**416**	792	792
inversion	**1,1** EV[1]	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	**1,1** EV[1]	1,0	1,0	**1,1** EV[1]	**1,1** EV[1]	1,0	1,0

5-19(01367)	**5-29(01368)**	**5-31(01369)**	**5-Z18(01457)**	**5-21(01458)**	**5-30(01468)**	**5-32(01469)**	**5-22(01478)**	**5-20(01568)**	**5-8(02346)**	**5-11(02347)**	**5-23(02357)**	**5-25(02358)**	**5-28(02368)**	**5-26(02458)**	**5-33(02468)**	**5-34(02469)**	**5-35(02479)**	**5-Z37(03458)**	
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**38**
5	5	5	5	5	5	5	3	5	3	5	5	5	5	5	3	3	3	3	**170**
15	15	15	15	15	15	15	9	15	9	15	15	15	15	15	9	9	9	9	**510**
35	35	35	35	35	35	35	19	35	19	35	35	35	35	35	19	19	19	19	**1170**
70	70	70	70	70	70	70	38	70	38	70	70	70	70	70	38	38	38	38	**2340**
126	126	126	126	126	126	126	66	126	66	126	126	126	126	126	66	66	66	66	**4188**
210	210	210	210	210	210	210	110	210	110	210	210	210	210	210	110	110	110	110	**6980**
330	330	330	330	330	330	330	170	330	170	330	330	330	330	330	170	170	170	170	**10940**
792	792	792	792	792	792	792	**416**	792	**416**	792	792	792	792	792	**416**	**416**	**416**	**416**	***26336***
1,0	1,0	1,0	1,0	1,0	1,0	1,0	**1,1** EV[1]	1,0	**1,1** EV[1]	1,0	1,0	1,0	1,0	1,0	**1,1** EV[1]	**1,1** EV[1]	**1,1** EV[1]	**1,1** EV[1]	

The Septachordal and Nonachordal Parent Classes

The septachord parent classes exhibit the same symmetry types as do their pentachord complements: 1,0; 1,1 EV^1. Likewise, the nonachord parent classes exhibit those of their trichord complements: 1,0; 1,1 EV^1; 3,3 EV^1. For the septachord parent classes, Figure 1.22 calculates the number of mscs for each symmetry type, and Figure 1.23 lists each septachord parent class with totals below for each number of elements in the multiset class. Similarly, for the nonachord parent classes, Figure 1.24 calculates the number of mscs for each symmetry type, and Figure 1.25 lists each nonachord parent with totals below for each number of elements in the multiset class.

The Tetrachordal and Octachordal Parent Classes

Enumeration of 4- and 8-object multiset classes continues, but here we encounter EV^2 symmetry. Parent classes with this symmetry type have a symmetrical axis running through two of its members. Each of these two members maps onto itself under this symmetry. In these cases we don't look for palindromic compositions but, instead, for what might be called *circular palindromes*. For example, the mfunction <1,1,2,1,1> can be seen as a palindromic composition. It may be applied to the parent 5-1(01234) to create a self-symmetrical offspring. (It may be rotated if necessary: <1,2,1,1,1> for 5-15(01268).) The mfunction <1,1,2,1,1,1>, however, is not palindromic no matter how it is rotated. Nevertheless it can be applied to a particular hexachordal parent, producing an inversionally self-symmetrical offspring: <1,1,2,1,1,1> applied to (012348) gives 0122348. This result is self-symmetrical: 0 and 1 map onto 4 and 3, respectively; the doubleton 22 maps onto itself, as does the singleton 8. I adopt here the term circular palindrome to refer to such compositions (mfunctions) because the circularity of pc space

Figure 1.22 Enumeration of the 7-object multiset classes. Symmetries: 1,0 and 1,1 EV[1]

	Partitions multiplicity of each object	**Permutations** (# of terms)!	multiplicity of multiplicity		mfunctions (compositions)	**1,0 symmetry**	*# of palindromic compositions*	**1,1 EV[1] symmetry**
7 elem.	1+1+1+1+1+1+1	7!	/ 7!		= 1	1	*1*	1
						1	*1*	**1**
8 elem.	2+1+1+1+1+1+1	7!	/ 6! * 1!		= 7	7	*1*	4
						7	*1*	**4**
9 elem.	3+1+1+1+1+1+1	7!	/ 6! * 1!		= 7	7	*1*	4
	2+2+1+1+1+1+1	7!	/ 5! * 2!		= 21	21	*3*	12
						28	*4*	**16**
10 elem.	4+1+1+1+1+1+1	7!	/ 6! * 1!		= 7	7	*1*	4
	3+2+1+1+1+1+1	7!	/ 5! * 1! * 1!		= 42	42	*0*	21
	2+2+2+1+1+1+1	7!	/ 4! * 3!		= 35	35	*3*	19
						84	*4*	**44**
11 elem.	5+1+1+1+1+1+1	7!	/ 6! * 1!		= 7	7	*1*	4
	4+2+1+1+1+1+1	7!	/ 5! * 1! * 1!		= 42	42	*0*	21
	3+3+1+1+1+1+1	7!	/ 5! * 2!		= 21	21	*3*	12
	3+2+2+1+1+1+1	7!	/ 4! * 2! * 1!		= 105	105	*3*	54
	2+2+2+2+1+1+1	7!	/ 4! * 3!		= 35	35	*3*	19
						210	*10*	**110**
12 elem.	6+1+1+1+1+1+1	7!	/ 6! * 1!		= 7	7	*1*	4
	5+2+1+1+1+1+1	7!	/ 5! * 1! * 1!		= 42	42	*0*	21
	4+3+1+1+1+1+1	7!	/ 5! * 1! * 1!		= 42	42	*0*	21
	4+2+2+1+1+1+1	7!	/ 4! * 2! * 1!		= 105	105	*3*	54
	3+3+2+1+1+1+1	7!	/ 4! * 2! * 1!		= 105	105	*3*	54
	3+2+2+2+1+1+1	7!	/ 3! * 3! * 1!		= 140	140	*0*	70
	2+2+2+2+2+1+1	7!	/ 5! * 2!		= 21	21	*3*	12
						462	*10*	**236**
				Totals:		**792**		**411**

Figure 1.23 Multiset Class Totals: Septachordal Parent Classes.

	7-1(0123456)	**7-2(0123457)**	**7-3(0123458)**	**7-4(0123467)**	**7-9(0123468)**	**7-10(0123469)**	**7-6(0123478)**	**7-Z12(0123479**	**7-5(0123567)**	**7-Z36(0123568**	**7-16(0123569)**	**7-14(0123578)**	**7-24(0123579)**	**7-7(0123678)**	**7-19(0123679)**	**7-13(0124568)**	**7-Z17(0124569**	**7-Z38(0124578**	**7-27(0124579)**
1 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
2 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
3 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
4 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
5 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
6 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
7 elem.	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
8 elem.	4	7	7	7	7	7	7	4	7	7	7	7	7	7	7	7	4	7	7
9 elem.	16	28	28	28	28	28	28	16	28	28	28	28	28	28	28	28	16	28	28
10 elem.	44	84	84	84	84	84	84	44	84	84	84	84	84	84	84	84	44	84	84
11 elem.	110	210	210	210	210	210	210	110	210	210	210	210	210	210	210	210	110	210	210
12 elem.	236	462	462	462	462	462	462	236	462	462	462	462	462	462	462	462	236	462	462
	411	792	792	792	792	792	792	**411**	792	792	792	792	792	792	792	792	**411**	792	792
	1,1	1,0	1,0	1,0	1,0	1,0	1,0	**1,1**	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	**1,1**	1,0	1,0
inversion	EV^1							EV^1									EV^1		

7-21(0124589)	**7-15(0124678)**	**7-29(0124679)**	**7-30(0124689)**	**7-33(012468T)**	**7-20(0125679)**	**7-22(0125689)**	**7-11(0134568)**	**7-Z37(0134578**	**7-26(0134579)**	**7-31(0134679)**	**7-32(0134689)**	**7-34(013468T)**	**7-28(0135679)**	**7-35(013568T)**	**7-Z18(0145679**	**7-8(0234568)**	**7-23(0234579)**	**7-25(0234679)**	
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**38**
7	4	7	7	4	7	4	7	4	7	7	7	4	7	4	7	4	7	7	**236**
28	16	28	28	16	28	16	28	16	28	28	28	16	28	16	28	16	28	28	**944**
84	44	84	84	44	84	44	84	44	84	84	84	44	84	44	84	44	84	84	**2792**
210	110	210	210	110	210	110	210	110	210	210	210	110	210	110	210	110	210	210	**6980**
462	236	462	462	236	462	236	462	236	462	462	462	236	462	236	462	236	462	462	**15296**
792	**411**	792	792	**411**	792	**411**	792	**411**	792	792	792	**411**	792	**411**	792	**411**	792	792	***26286***
1,0	**1,1**	1,0	1,0	**1,1**	1,0	**1,1**	1,0	**1,1**	1,0	1,0	1,0	**1,1**	1,0	**1,1**	1,0	**1,1**	1,0	1,0	
	EV^1			EV^1		EV^1		EV^1				EV^1		EV^1		EV^1			

Figure 1.24 Enumeration of the 9-object multiset classes. Symmetries: 1,0 and 1,1 EV^1

	Partitions multiplicity of each object	**Permutations** (# of terms)!	multiplicity of multiplicity		mfunctions (compositions)	**1,0 symmetry**	*# of palindromic compositions*	**1,1 EV^1 symmetry**	*# of compositions with 3 degrees of T-symm*	**3,3 EV^1 symmetry**
9 elem.	1+1+1+1+1+1+1+1+1	9!	/ 9!	= 1		1	*1*	1	*1*	1
						1		**1**		**1**
10 elem.	2+1+1+1+1+1+1+1+1	9!	/ 8! * 1!	= 9		9	*1*	5	*0*	2
						9		**5**		**2**
11 elem.	3+1+1+1+1+1+1+1+1	9!	/ 8! * 1!	= 9		9	*1*	5	*0*	2
	2+2+1+1+1+1+1+1+1	9!	/ 7! * 2!	= 36		36	*4*	20	*0*	8
						45		**25**		**10**
12 elem.	4+1+1+1+1+1+1+1+1	9!	/ 8! * 1!	= 9		9	*1*	5	*0*	2
	3+2+1+1+1+1+1+1+1	9!	/ 7! * 1! * 1!	= 72		72	*0*	36	*0*	12
	2+2+2+1+1+1+1+1+1	9!	/ 6! * 3!	= 84		84	*4*	44	*3*	17
						165	*5*	**85**	*3*	**31**
				Totals:		**220**		**116**		**44**

Figure 1.25 Multiset Class Totals: Nonachordal Parent Classes.

	9-1(012345678)	**9-2(012345679)**	**9-3(012345689)**	**9-6(01234568T)**	**9-4(012345789)**	**9-7(01234578T)**	**9-5(012346789)**	**9-8(01234678T)**	**9-10(01234679T)**	**9-9(01235678T)**	**9-11(01235679T)**	**9-12(01245689T)**	
1 elem.	x	x	x	x	x	x	x	x	x	x	x	x	
2 elem.	x	x	x	x	x	x	x	x	x	x	x	x	
3 elem.	x	x	x	x	x	x	x	x	x	x	x	x	
4 elem.	x	x	x	x	x	x	x	x	x	x	x	x	
5 elem.	x	x	x	x	x	x	x	x	x	x	x	x	
6 elem.	x	x	x	x	x	x	x	x	x	x	x	x	
7 elem.	x	x	x	x	x	x	x	x	x	x	x	x	
8 elem.	x	x	x	x	x	x	x	x	x	x	x	x	
9 elem.	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**12**
10 elem.	5	9	9	5	9	9	9	9	5	5	9	2	**85**
11 elem.	25	45	45	25	45	45	45	45	25	25	45	10	**425**
12 elem.	85	165	165	85	165	165	165	165	85	85	165	31	**1526**
	116	220	220	**116**	220	220	220	220	**116**	**116**	220	**44**	***2048***
symmetry	**1,1 EV^1**	1,0	1,0	**1,1 EV^1**	1,0	1,0	1,0	1,0	**1,1 EV^1**	**1,1 EV^1**	1,0	**3,3 EV^1**	

Is required for the palindrome to be in effect. Such mfunctions are essentially palindromic mfunctions (with an odd number of positions) with an "extra" position at the end. While the axis runs through the position at the center of the palindrome, it also runs through this "extra" position:

Offspring from parent classes of the symmetry type 1,1 EV^2, then, can be calculated with Burnside's Lemma, using number of compositions (mfunctions) for the T_0 position and using number of circular palindromes for the I position. The lower right of figure 1.26a shows exactly this. (I* represents number or circular palindromic compositions, while I represents the standard palindromes for ODD, EV^0, and EV^1 symmetry types.) At the lower left of the figure is the calculation of the eleven 8-element offspring of a parent with 2,2 EV^0 or 2,2 ODD symmetry. It uses number of compositions (mfunctions) and number of palindromic compositions, but this calculation also uses number of compositions with 2 degrees of T-symm. As before, we identify mfunctions with T-symmetry by examining their MM values. If each term in the multiplicity of multiplicity is divisible by two, then the partition may yield an mfunction or mfunctions with not just T_0 symmetry, but also T_6 symmetry. So where the following determines the number of mfunctions for a 3+3+1+1 partition,

$$\frac{4!}{2! * 2!} = 6$$

the following (each term divided by 2) determines the number of mfunctions with 2 degrees of symmetry:

$$\frac{2!}{1! * 1!} = 2$$

These results, 6 and 2, can be found in Figure 1.26a. The number of compositions with 2 degrees of symmetry will be the same as the number of palindromic compostions, because each type is an mfunction (and resulting multiset class) segmented into two halves, each half mapping completely onto the other. One maps by transposition, the other by inversion: <3,1,1,3> is a palindromic composition, and <3,1,3,1> is a composition with 2 degrees of symmetry. The latter is a palindromic composition with its back half retrograded.

Figure 1.26b elaborates on 4,4 B symmetry. The B represents Alegant's term, "Both." This applied to those set classes with both EVEN and ODD inversional symmetry. They are the set classes 4-28 (0369), its complement 8-28, and the aggregate itself. Specifically, 4-28(0369) exhibits EV^2 and ODD symmetry. (EV^2 is found when, say, 0 and 6 each map to themselves, and 3 and 6 map to each other. ODD is found when the members pair off: 3 onto 6 and 0 onto 9.) We once again use Burnside's Lemma, but there will be eight entries, one for each degree of symmetry: (ψT_0) = number of mfunctions; (ψT_3) and (ψT_9) = number of compositions with 4 degrees of T-symm; (ψT_6) = number of compositions with 6 degrees of T-symm; (ψI) = number of palindromic compositions; and (ψI^*) = number of circular palindromes.

Figure 1.26a Calculating totals of the 8-element offspring of the canonical tetrachords with 2,2 EV^0; 1,1 ODD; and 1,1 EV^2 symmetries.

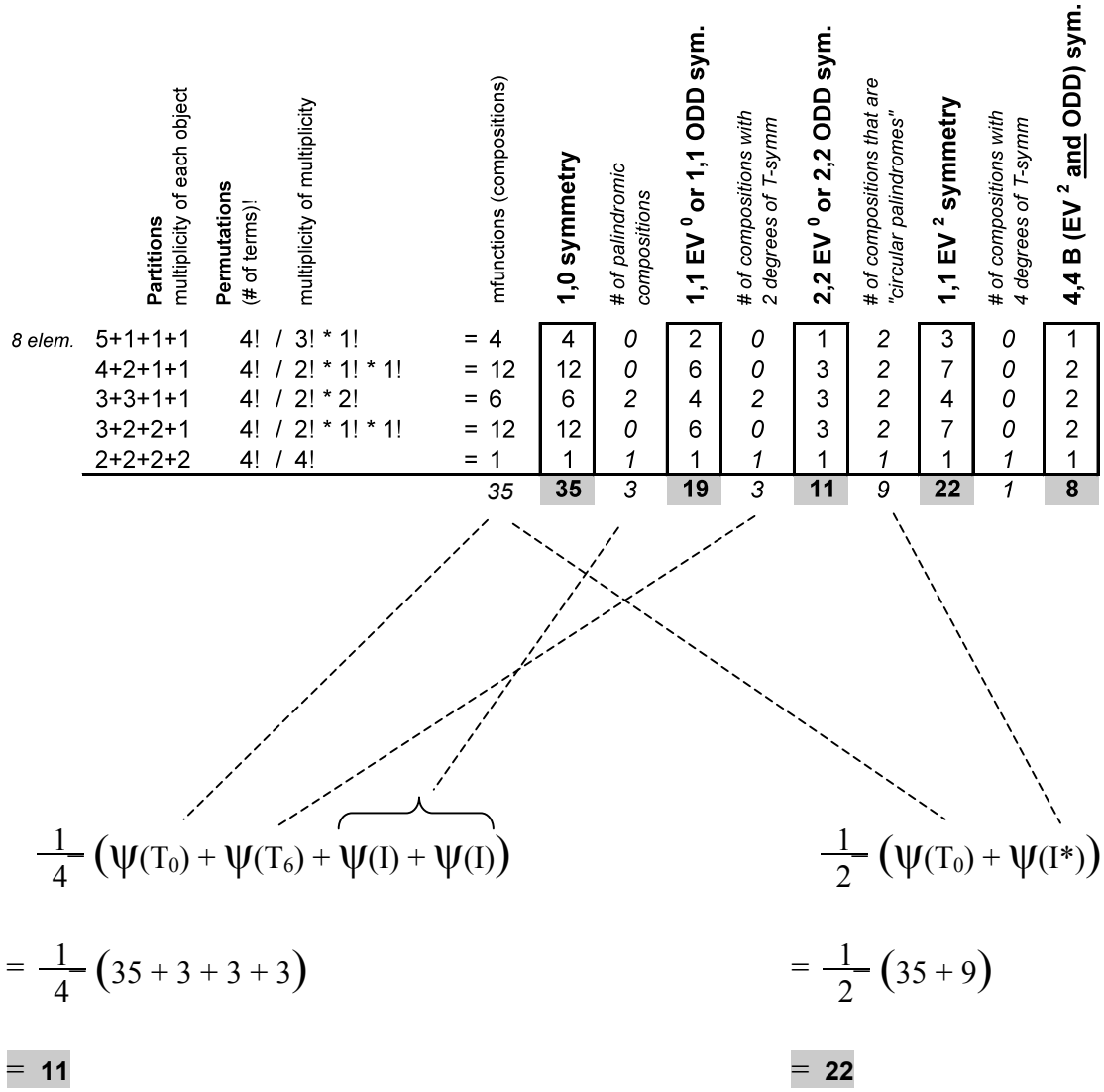


Figure 1.26b Calculating totals of the 8-element offspring of the canonical tetrachords with 4,4 B symmetry

	Partitions multiplicity of each object	**Permutations** (# of terms)! / multiplicity of multiplicity	mfunctions (compositions)	**1,0 symmetry**	*# of palindromic compositions*	**1,1 EV ^{0} or 1,1 ODD sym.**	*# of compositions with 2 degrees of T-symm*	**2,2 EV ^{0} or 2,2 ODD sym.**	*# of compositions that are "circular palindromes"*	**1,1 EV ^{2} symmetry**	*# of compositions with 4 degrees of T-symm*	**4,4 B (EV ^{2} and ODD) sym.**
8 elem.	5+1+1+1	4! / 3! * 1!	= 4	4	*0*	2	*0*	1	*2*	3	*0*	1
	4+2+1+1	4! / 2! * 1! * 1!	= 12	12	*0*	6	*0*	3	*2*	7	*0*	2
	3+3+1+1	4! / 2! * 2!	= 6	6	*2*	4	*2*	3	*2*	4	*0*	2
	3+2+2+1	4! / 2! * 1! * 1!	= 12	12	*0*	6	*0*	3	*2*	7	*0*	2
	2+2+2+2	4! / 4!	= 1	1	*1*	1	*1*	1	*1*	1	*1*	1
			35	**35**	*3*	**19**	*3*	**11**	*9*	**22**	*1*	**8**

$$\frac{1}{8}\left(\psi(T_0) + \psi(T_3) + \psi(T_6) + \psi(T_9) + \psi(I) + \psi(I) + \psi(I^*) + \psi(I^*)\right)$$

$$= \frac{1}{8}\left(35 + 1 + 3 + 1 + 3 + 3 + 9 + 9\right)$$

$$= \mathbf{8}$$

Figure 1.27 carries out the above calculations up to 12 elements, and Figure 1.28 distributes these subtotals to each tetrachordal parent. Figures 1.29 and 1.30 repeat the preceding procedure for the 8-object multiset classes (octachordal parent classes).

Figure 1.27 Enumeration of the 4-object multiset classes. Symmetries: 1,0; 1,1 EV^0; 1,1 ODD; 2,2 EV^0; 2,2 ODD; 1,1 EV^2; 4,4 B (EV^2 and ODD).

	Partitions multiplicity of each object	Permutations (# of terms)!	multiplicity of multiplicity	mfunctions (compositions)	**1,0 symmetry**	*# of palindromic compositions*	**1,1 EV^0 or 1,1 ODD sym.**	*# of compositions with 2 degrees of T-symm*	**2,2 EV^0 or 2,2 ODD sym.**	*# of compositions that are "circular palindromes"*	**1,1 EV^2 symmetry**	*# of compositions with 4 degrees of T-symm*	**4,4 B (EV^2 and ODD) sym.**
4 elem.	1+1+1+1	4!	/ 4!	= 1	1	*1*	1	*1*	1	*1*	1	*1*	1
					1	*1*	**1**	*1*	**1**	*1*	**1**	*1*	**1**
5 elem.	2+1+1+1	4!	/ 3! * 1!	= 4	4	*0*	2	*0*	1	*2*	3	*0*	1
					4	*0*	**2**	*0*	**1**	*2*	**3**	*0*	**1**
6 elem.	3+1+1+1	4!	/ 3! * 1!	= 4	4	*0*	2	*0*	1	*2*	3	*0*	1
	2+2+1+1	4!	/ 2! * 2!	= 6	6	*2*	4	*2*	3	*2*	4	*0*	2
					10	*2*	**6**	*2*	**4**	*4*	**7**	*0*	**3**
7 elem.	4+1+1+1	4!	/ 3! * 1!	= 4	4	*0*	2	*0*	1	*2*	3	*0*	1
	3+2+1+1	4!	/ 2! * 1! * 1!	= 12	12	*0*	6	*0*	3	*2*	7	*0*	2
	2+2+2+1	4!	/ 3! * 1!	= 4	4	*0*	2	*0*	1	*2*	3	*0*	1
					20	*0*	**10**	*0*	**5**	*6*	**13**	*0*	**4**
8 elem.	5+1+1+1	4!	/ 3! * 1!	= 4	4	*0*	2	*0*	1	*2*	3	*0*	1
	4+2+1+1	4!	/ 2! * 1! * 1!	= 12	12	*0*	6	*0*	3	*2*	7	*0*	2
	3+3+1+1	4!	/ 2! * 2!	= 6	6	*2*	4	*2*	3	*2*	4	*0*	2
	3+2+2+1	4!	/ 2! * 1! * 1!	= 12	12	*0*	6	*0*	3	*2*	7	*0*	2
	2+2+2+2	4!	/ 4!	= 1	1	*1*	1	*1*	1	*1*	1	*1*	1
					35	*3*	**19**	*3*	**11**	*9*	**22**	*1*	**8**
9 elem.	6+1+1+1	4!	/ 3! * 1!	= 4	4	*0*	2	*0*	1	*2*	3	*0*	1
	5+2+1+1	4!	/ 2! * 1! * 1!	= 12	12	*0*	6	*0*	3	*2*	7	*0*	2
	4+3+1+1	4!	/ 2! * 1! * 1!	= 12	12	*0*	6	*0*	3	*2*	7	*0*	2
	4+2+2+1	4!	/ 2! * 1! * 1!	= 12	12	*0*	6	*0*	3	*2*	7	*0*	2
	3+3+2+1	4!	/ 2! * 1! * 1!	= 12	12	*0*	6	*0*	3	*2*	7	*0*	2
	3+2+2+2	4!	/ 3! * 1!	= 4	4	*0*	2	*0*	1	*2*	3	*0*	1
					56	*0*	**28**	*0*	**14**	*12*	**34**	*0*	**10**

Figure 1.27 (continued)

	Partitions multiplicity of each object	Permutations (# of terms)!	multiplicity of multiplicity		mfunctions (compositions)	**1,0 symmetry**	*# of palindromic compositions*	**1,1 EV ^{0} or 1,1 ODD sym.**	*# of compositions with 2 degrees of T-symm*	**2,2 EV ^{0} or 2,2 ODD sym.**	*# of compositions that are "circular palindromes"*	**1,1 EV ^{2} symmetry**	*# of compositions with 4 degrees of T-symm*	**4,4 B (EV ^{2} and ODD) sym.**
10 elem.	7+1+1+1	4! /	3! * 1!	= 4		4	*0*	2	*0*	1	*2*	3	*0*	1
	6+2+1+1	4! /	2! * 1! * 1!	= 12		12	*0*	6	*0*	3	*2*	7	*0*	2
	5+3+1+1	4! /	2! * 1! * 1!	= 12		12	*0*	6	*0*	3	*2*	7	*0*	2
	5+2+2+1	4! /	2! * 1! * 1!	= 12		12	*0*	6	*0*	3	*2*	7	*0*	2
	4+4+1+1	4! /	2! * 2!	= 6		6	*2*	4	*2*	3	*2*	4	*0*	2
	4+3+2+1	4! /	1! * 1! * 1! * 1!	= 24		24	*0*	12	*0*	6	*0*	12	*0*	3
	4+2+2+2	4! /	3! * 1!	= 4		4	*0*	2	*0*	1	*2*	3	*0*	1
	3+3+3+1	4! /	3! * 1!	= 4		4	*0*	2	*0*	1	*2*	3	*0*	1
	3+3+2+2	4! /	2! * 2!	= 6		6	*2*	4	*2*	3	*2*	4	*0*	2
						84	*4*	**44**	*4*	**24**	*16*	**50**	*0*	**16**
11 elem.	8+1+1+1	4! /	3! * 1!	= 4		4	*0*	2	*0*	1	*2*	3	*0*	1
	7+2+1+1	4! /	2! * 1! * 1!	= 12		12	*0*	6	*0*	3	*2*	7	*0*	2
	6+3+1+1	4! /	2! * 1! * 1!	= 12		12	*0*	6	*0*	3	*2*	7	*0*	2
	6+2+2+1	4! /	2! * 1! * 1!	= 12		12	*0*	6	*0*	3	*2*	7	*0*	2
	5+4+1+1	4! /	2! * 1! * 1!	= 12		12	*0*	6	*0*	3	*2*	7	*0*	2
	5+3+2+1	4! /	1! * 1! * 1! * 1!	= 24		24	*0*	12	*0*	6	*0*	12	*0*	3
	5+2+2+2	4! /	3! * 1!	= 4		4	*0*	2	*0*	1	*2*	3	*0*	1
	4+4+2+1	4! /	2! * 1! * 1!	= 12		12	*0*	6	*0*	3	*2*	7	*0*	2
	4+3+3+1	4! /	2! * 1! * 1!	= 12		12	*0*	6	*0*	3	*2*	7	*0*	2
	4+3+2+2	4! /	2! * 1! * 1!	= 12		12	*0*	6	*0*	3	*2*	7	*0*	2
	3+3+3+2	4! /	3! * 1!	= 4		4	*0*	2	*0*	1	*2*	3	*0*	1
						120	*0*	**60**	*0*	**30**	*20*	**70**	*0*	**20**
12 elem.	9+1+1+1	4! /	3! * 1!	= 4		4	*0*	2	*0*	1	*2*	3	*0*	1
	8+2+1+1	4! /	2! * 1! * 1!	= 12		12	*0*	6	*0*	3	*2*	7	*0*	2
	7+3+1+1	4! /	2! * 1! * 1!	= 12		12	*0*	6	*0*	3	*2*	7	*0*	2
	7+2+2+1	4! /	2! * 1! * 1!	= 12		12	*0*	6	*0*	3	*2*	7	*0*	2
	6+4+1+1	4! /	2! * 1! * 1!	= 12		12	*0*	6	*0*	3	*2*	7	*0*	2
	6+3+2+1	4! /	1! * 1! * 1! * 1!	= 24		24	*0*	12	*0*	6	*0*	12	*0*	3
	6+2+2+2	4! /	3! * 1!	= 4		4	*0*	2	*0*	1	*2*	3	*0*	1
	5+5+1+1	4! /	2! * 2!	= 6		6	*2*	4	*2*	3	*2*	4	*0*	2
	5+4+2+1	4! /	1! * 1! * 1! * 1!	= 24		24	*0*	12	*0*	6	*0*	12	*0*	3
	5+3+3+1	4! /	2! * 1! * 1!	= 12		12	*0*	6	*0*	3	*2*	7	*0*	2
	5+3+2+2	4! /	2! * 1! * 1!	= 12		12	*0*	6	*0*	3	*2*	7	*0*	2
	4+4+3+1	4! /	2! * 1! * 1!	= 12		12	*0*	6	*0*	3	*2*	7	*0*	2
	4+4+2+2	4! /	2! * 2!	= 6		6	*2*	4	*2*	3	*2*	4	*0*	2
	4+3+3+2	4! /	2! * 1! * 1!	= 12		12	*0*	6	*0*	3	*2*	7	*0*	2
	3+3+3+3	4! /	4!	= 1		1	*1*	1	*1*	1	*1*	1	*1*	1
						165	*5*	**85**	*5*	**45**	*25*	**95**	*1*	**29**
				Totals:		**495**		**255**		**135**		**295**		**92**

Figure 1.28 Multiset Class Totals: Tetrachordal Parent Classes

	4-1(0123)	4-2(0124)	4-4(0125)	4-5(0126)	4-6(0127)	4-3(0134)	4-11(0135)	4-13(0136)	4-Z29(0137)	4-7(0145)	4-Z15(0146)	4-18(0147)	4-19(0148)	4-8(0156)
1 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x
2 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x
3 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x
4 elem.	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
5 elem.	2	4	4	4	3	2	4	4	4	2	4	4	4	2
6 elem.	6	10	10	10	7	6	10	10	10	6	10	10	10	6
7 elem.	10	20	20	20	13	10	20	20	20	10	20	20	20	10
8 elem.	19	35	35	35	22	19	35	35	35	19	35	35	35	19
9 elem.	28	56	56	56	34	28	56	56	56	28	56	56	56	28
10 elem.	44	84	84	84	50	44	84	84	84	44	84	84	84	44
11 elem.	60	120	120	120	70	60	120	120	120	60	120	120	120	60
12 elem.	85	165	165	165	95	85	165	165	165	85	165	165	165	85
	255	495	495	495	295	255	495	495	495	255	495	495	495	255
symmetry	1,1	1,0	1,0	1,0	1,1	1,1	1,0	1,0	1,0	1,1	1,0	1,0	1,0	1,1
inversion	ODD				EV^2	EV^0				ODD				EV^0

4-16(0157)	4-20(0158)	4-9(0167)	4-10(0235)	4-12(0236)	4-14(0237)	4-21(0246)	4-22(0247)	4-24(0248)	4-23(0257)	4-27(0258)	4-25(0268)	4-17(0347)	4-26(0358)	4-28(0369)	
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**29**
4	2	1	2	4	4	2	4	3	2	4	1	2	2	1	**85**
10	6	4	6	10	10	6	10	7	6	10	4	6	6	3	**225**
20	10	5	10	20	20	10	20	13	10	20	5	10	10	4	**420**
35	19	11	19	35	35	19	35	22	19	35	11	19	19	8	**754**
56	28	14	28	56	56	28	56	34	28	56	14	28	28	10	**1170**
84	44	24	44	84	84	44	84	50	44	84	24	44	44	16	**1780**
120	60	30	60	120	120	60	120	70	60	120	30	60	60	20	**2500**
165	85	45	85	165	165	85	165	95	85	165	45	85	85	29	**3469**
495	255	135	255	495	495	255	495	295	255	495	135	255	255	92	***10432***
1,0	1,1	2,2	1,1	1,0	1,0	1,1	1,0	1,1	1,1	1,0	2,2	1,1	1,1	4,4	
	ODD	ODD	ODD			EV^0		EV^2	ODD		EV^0	ODD	EV^0	B	

Figure 1.29 Enumeration of the 8-object multiset classes. Symmetries: 1,0; 1,1 EV^0; 1,1 ODD; 2,2 ODD; 1,1 EV^2; 2,2 EV^2; 4,4 B (EV^0 and ODD).

	Partitions multiplicity of each object	Permutations (# of terms)! / multiplicity of multiplicity	= functions (compositions)	**1,0 symmetry**	*# of palindromic compositions*	**1,1 EV^0 or 1,1 ODD sym.**	*# of compositions with 2 degrees of T-symm*	**2,2 ODD sym.**	*# of compositions that are "circular palindromes"*	**1,1 EV^2 symmetry**	**2,2 EV^2 symmetry**	*# of compositions with 4 degrees of T-symm*	**4,4 B (EV^0 and ODD) sym.**
8 elem.	1+1+1+1+1+1+1+1	8! / 8!	= 1	1	*1*	1	*1*	1	*1*	1	1	*1*	1
				1	*1*	**1**	*1*	**1**	*1*	**1**	**1**	*1*	**1**
9 elem.	2+1+1+1+1+1+1+1	8! / 7! * 1!	= 8	8	*0*	4	*0*	2	*2*	5	3	*0*	1
				8	*0*	**4**	*0*	**2**	*2*	**5**	**3**	*0*	**1**
10 elem.	3+1+1+1+1+1+1+1	8! / 7! * 1!	= 8	8	*0*	4	*0*	2	*2*	5	3	*0*	1
	2+2+1+1+1+1+1+1	8! / 6! * 2!	= 28	28	*4*	16	*4*	10	*4*	16	10	*0*	6
				36	*4*	**20**	*4*	**12**	*6*	**21**	**13**	*0*	**7**
11 elem.	4+1+1+1+1+1+1+1	8! / 7! * 1!	= 8	8	*0*	4	*0*	2	*2*	5	3	*0*	1
	3+2+1+1+1+1+1+1	8! / 6! * 1! * 1!	= 56	56	*0*	28	*0*	14	*2*	29	15	*0*	7
	2+2+2+1+1+1+1+1	8! / 5! * 3!	= 56	56	*0*	28	*0*	14	*6*	31	17	*0*	7
				120	*0*	**60**	*0*	**30**	*10*	**65**	**35**	*0*	**15**
12 elem.	5+1+1+1+1+1+1+1	8! / 7! * 1!	= 8	8	*0*	4	*0*	2	*2*	5	3	*0*	1
	4+2+1+1+1+1+1+1	8! / 6! * 1! * 1!	= 56	56	*0*	28	*0*	14	*2*	29	15	*0*	7
	3+3+1+1+1+1+1+1	8! / 6! * 2!	= 28	28	*4*	16	*4*	10	*4*	16	10	*0*	6
	3+2+2+1+1+1+1+1	8! / 5! * 2! *1!	= 168	168	*0*	84	*0*	42	*6*	87	45	*0*	21
	2+2+2+2+1+1+1+1	8! / 4! * 4!	= 70	70	*6*	38	*6*	22	*6*	38	22	*2*	13
				330	*10*	**170**	*10*	**90**	*20*	**175**	**95**	*2*	**48**
			Totals:	**495**		**255**		**135**		**267**	**147**		**72**

Figure 1.30 Multiset Class Totals: Octachordal Parent Classes.

	8-1(01234567)	8-2(01234568)	8-3(01234569)	8-4(01234578)	8-11(01234579)	8-7(01234589)	8-5(01234678)	8-13(01234679)	8-Z15(01234689)	8-21(0123468T)	8-8(01234789)	8-6(01235678)	8-Z29(01235679)	8-18(01235689)	8-22(0123568T)
1 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
2 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
3 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
4 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
5 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
6 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
7 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
8 elem.	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
9 elem.	4	8	5	8	8	4	8	8	8	5	5	4	8	8	8
10 elem.	20	36	21	36	36	20	36	36	36	21	21	20	36	36	36
11 elem.	60	120	65	120	120	60	120	120	120	65	65	60	120	120	120
12 elem.	170	330	175	330	330	170	330	330	330	175	175	170	330	330	330
	255	495	267	495	495	255	495	495	495	267	267	255	495	495	495
symmetry	1,1	1,0	1,1	1,0	1,0	1,1	1,0	1,0	1,0	1,1	1,1	1,1	1,0	1,0	1,0
inversion	ODD		EV^2			ODD				EV^2	EV^2	EV^0			

	8-16(0123789)	8-23(0123578T)	8-9(0123679)	8-14(0124567)	8-19(0124568)	8-24(0124568T)	8-20(0124578)	8-27(0124578T)	8-25(0124678T)	8-12(0134567)	8-17(0134568)	8-26(0134578T)	8-28(0134679T)	8-10(0234567)	
	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
	1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**29**
	8	4	2	8	8	4	4	8	3	8	4	5	1	4	**170**
	36	20	12	36	36	20	20	36	13	36	20	21	7	20	**780**
	120	60	30	120	120	60	60	120	35	120	60	65	15	60	**2500**
	330	170	90	330	330	170	170	330	95	330	170	175	48	170	**6913**
	495	255	135	495	495	255	255	495	147	495	255	267	72	255	***10392***
	1,0	1,1	2,2	1,0	1,0	1,1	1,1	1,0	2,2	1,0	1,1	1,1	4,4	1,1	
		ODD	ODD			EV^0	ODD		EV^2		ODD	EV^2	B	ODD	

The Hexachordal Parent Classes

The hexachordal parent classes bring yet another wrinkle. Parent classes 6-7(012678) and 6-35(02468T) have six degrees of transpositional symmetry, but they have two different types of EVEN inversional symmetry (EV^0 and EV^2) at three degrees each. In these cases we use the notation, 6,6 $EV^{0\text{-}2}$. Alegant's term "BOTH" does not apply because it is not EVEN and ODD symmetry here, only EVEN. Figure 1.31 shows how the 12 entries in Burnside's Lemma are determined. There are three (I) entries representing the EV^0-symmetrical palindromes and three (I*) entries representing the EV^2-symmetric circular palindromes. Figure 1.32 carries out the operations up to 12 elements, and Figure 1.33 compiles the totals for each parent class.

The Dyadic and Decachordal Parent Classes

The 2- and 10-object multiset classes give us our first encounter with cardinalites of parent classes with no 1,0 symmetry. That is, no dyads or decachords are *not* inversionally symmetrical at some degree. Figures 1.34 to 1.37 show the enumeration and compilation of total multiset offspring for these parent classes.

The Monadic, Hendecachordal and Dodecachordal Parent Classes

These somewhat trivial collections are enumerated and their totals are compiled in Figures 1.38 and 1.39. The enumeration of multiset classes off-sprung from sc 12-1(0123456789TE) with 12,12 B symmetry is featured at the bottom of Figure 1.38.

After having accounted for transpositional and inversional equivalence of multiset classes, I am now able to provide a complete accounting of multiset classes of with up to

twelve objects and up to twelve elements. The table in Figure 1.40, essentially a reconstruction of Figure 1.11 but without inversional redundancies, shows all subtotals as well as the grand total of 113,973 multiset classes. One finds complementary patterning—29 tetrachords and 29 octachords; 12 trichords and 12 nonachords—in the enumeration of the set classes, but there is no analogous patterning here. None, that is, that I have yet discovered.

Figure 1.31 Calculating totals of the 8-element offspring of the canonical hexachords with 6-6 EV^{0-2} symmetry

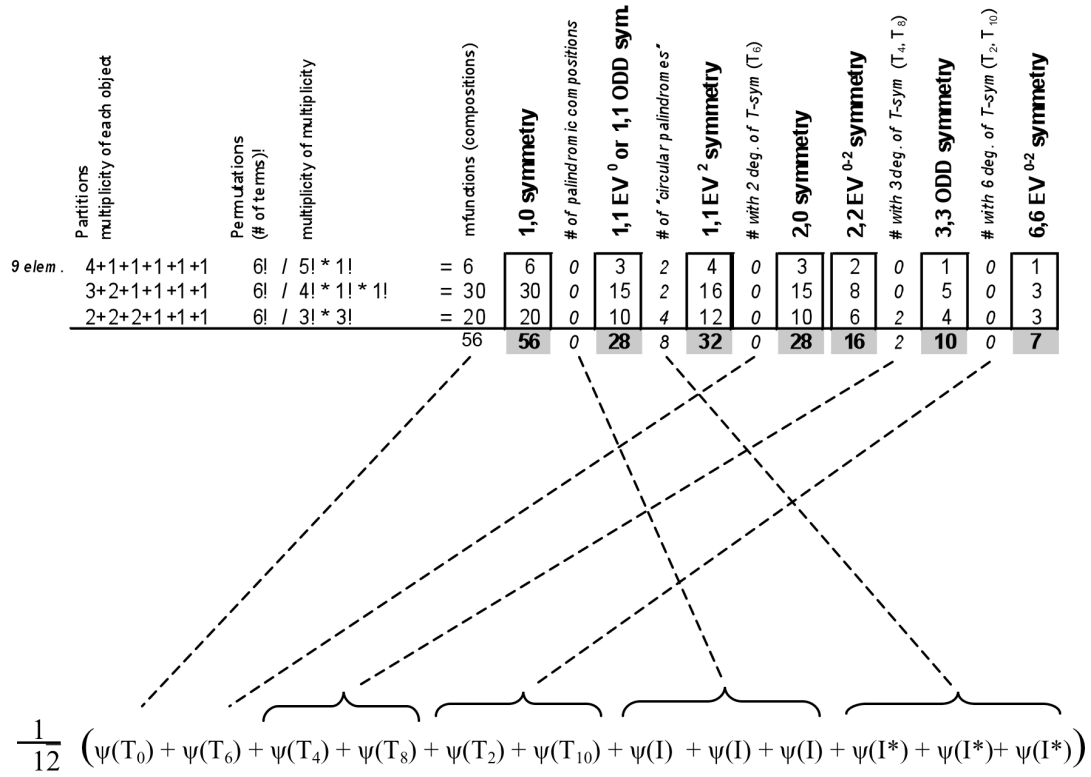


Figure 1.32 Enumeration of the 6-object multiset classes. Symmetries: 1,0; 1,1 EV^0; 1,1 ODD; 1,1 EV^2; 2,0 (T-only); 2,2 $EV^{0\text{-}2}$; 3,3 ODD; 6,6 $EV^{0\text{-}2}$.

	Partitions multiplicity of each object	Permutations (# of terms)! / multiplicity of multiplicity	mfunctions (compositions)	1,0 symmetry	# of palindromic compositions	1,1 EV^0 or 1,1 ODD sym.	# of "circular palindromes"	1,1 EV^2 symmetry	# with 2 deg. of T-sym (T_6)	2,0 symmetry	2,2 $EV^{0\text{-}2}$ symmetry	# with 3 deg. of T-sym (T_4, T_8)	3,3 ODD symmetry	# with 6 deg. of T-sym (T_2, T_{10})	6,6 $EV^{0\text{-}2}$ symmetry
6 elem.	1+1+1+1+1+1	6! / 6!	= 1	1	1	1	1	1	1	1	1	1	1	1	1
				1	1	**1**	1	**1**	1	**1**	**1**	1	**1**	1	**1**
7 elem.	2+1+1+1+1+1	6! / 5! * 1!	= 6	6	0	3	2	4	0	3	2	0	1	0	1
				6	0	**3**	2	**4**	0	**3**	**2**	0	**1**	0	**1**
8 elem.	3+1+1+1+1+1	6! / 5! * 1!	= 6	6	0	3	2	4	0	3	2	0	1	0	1
	2+2+1+1+1+1	6! / 4! * 2!	= 15	15	3	9	3	9	3	9	6	0	4	0	3
				21	3	**12**	3	**13**	3	**12**	**8**	0	**5**	0	**4**
9 elem.	4+1+1+1+1+1	6! / 5! * 1!	= 6	6	0	3	2	4	0	3	2	0	1	0	1
	3+2+1+1+1+1	6! / 4! * 1! * 1!	= 30	30	0	15	2	16	0	15	8	0	5	0	3
	2+2+2+1+1+1	6! / 3! * 3!	= 20	20	0	10	4	12	0	10	6	2	4	0	3
				56	0	**28**	8	**32**	0	**28**	**16**	2	**10**	0	**7**
10 elem.	5+1+1+1+1+1	6! / 5! * 1!	= 6	6	0	3	2	4	0	3	2	0	1	0	1
	4+2+1+1+1+1	6! / 4! * 1! * 1!	= 30	30	0	15	2	16	0	15	8	0	5	0	3
	3+3+1+1+1+1	6! / 4! * 2!	= 15	15	3	9	3	9	3	9	6	0	4	0	3
	3+2+2+1+1+1	6! / 3! * 2! * 1!	= 60	60	0	30	4	32	0	30	16	0	10	0	6
	2+2+2+2+1+1	6! / 4! * 2!	= 15	15	3	9	3	9	3	9	6	0	4	0	3
				126	6	**66**	14	**70**	6	**66**	**38**	0	**24**	0	**16**
11 elem.	6+1+1+1+1+1	6! / 5! * 1!	= 6	6	0	3	2	4	0	3	2	0	1	0	1
	5+2+1+1+1+1	6! / 4! * 1! * 1!	= 30	30	0	15	2	16	0	15	8	0	5	0	3
	4+3+1+1+1+1	6! / 4! * 1! * 1!	= 30	30	0	15	2	16	0	15	8	0	5	0	3
	4+2+2+1+1+1	6! / 3! * 2! * 1!	= 60	60	0	30	4	32	0	30	16	0	10	0	6
	3+3+2+1+1+1	6! / 3! * 2! * 1!	= 60	60	0	30	4	32	0	30	16	0	10	0	6
	3+2+2+2+1+1	6! / 3! * 2! * 1!	= 60	60	0	30	4	32	0	30	16	0	10	0	6
	2+2+2+2+2+1	6! / 5! * 1!	= 6	6	0	3	2	4	0	3	2	0	1	0	1
				252	0	**126**	20	**136**	0	**126**	**68**	0	**42**	0	**26**
12 elem.	7+1+1+1+1+1	6! / 5! * 1!	= 6	6	0	3	2	4	0	3	2	0	1	0	1
	6+2+1+1+1+1	6! / 4! * 1! * 1!	= 30	30	0	15	2	16	0	15	8	0	5	0	3
	5+3+1+1+1+1	6! / 4! * 1! * 1!	= 30	30	0	15	2	16	0	15	8	0	5	0	3
	5+2+2+1+1+1	6! / 3! * 2! * 1!	= 60	60	0	30	4	32	0	30	16	0	10	0	6
	4+4+1+1+1+1	6! / 4! * 2!	= 15	15	3	9	3	9	3	9	6	0	4	0	3
	4+3+2+1+1+1	6! / 3! * 1! * 1! * 1!	= 120	120	0	60	0	60	0	60	30	0	20	0	10
	4+2+2+2+1+1	6! / 3! * 2! * 1!	= 60	60	0	30	4	32	0	30	16	0	10	0	6
	3+3+3+1+1+1	6! / 3! * 3!	= 20	20	0	10	4	12	0	10	6	2	4	0	3
	3+3+2+2+1+1	6! / 2! * 2! * 2!	= 90	90	6	48	6	48	6	48	27	0	18	0	11
	3+2+2+2+2+1	6! / 4! * 1! * 1!	= 30	30	0	15	2	16	0	15	8	0	5	0	3
	2+2+2+2+2+2	6! / 6!	= 1	1	1	1	1	1	1	1	1	1	1	1	1
				462	10	**236**	30	**246**	10	**236**	**128**	3	**83**	1	**50**
		Totals:		**924**		**472**		**504**		**472**	**261**		**166**		**105**

Figure 1.33 Multiset Class Totals: Hexachordal Parent Classes.

	6-1(012345)	6-2(012346)	6-Z36(012347)	6-Z37(012348)	6-Z3(012356)	6-9(012357)	6-Z40(012358)	6-5(012367)	6-Z41(012368)	6-Z42(012369)	6-Z38(012378)	6-Z4(012456)	6-Z11(012457)	6-15(012458)	6-Z12(012467)	6-22(012468)	6-Z46(012469)	6-Z17(012478)	6-Z47(012479)	6-Z6(012567)	6-Z43(012568)	6-Z44(012569)	6-18(012578)	6-Z48(012579)	6-7(012678)
1 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
2 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
3 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
4 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
5 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
6 elem.	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
7 elem.	3	6	6	4	6	6	6	6	6	3	3	3	6	6	6	6	6	6	6	3	6	6	6	4	2
8 elem.	12	21	21	13	21	21	21	21	21	12	12	12	21	21	21	21	21	21	21	12	21	21	21	13	8
9 elem.	28	56	56	32	56	56	56	56	56	28	28	28	56	56	56	56	56	56	56	28	56	56	56	32	16
10 elem.	66	126	126	70	126	126	126	126	126	66	66	66	126	126	126	126	126	126	126	66	126	126	126	70	38
11 elem.	126	252	252	136	252	252	252	252	252	126	126	126	252	252	252	252	252	252	252	126	252	252	252	136	68
12 elem.	236	462	462	246	462	462	462	462	462	236	236	236	462	462	462	462	462	462	462	236	462	462	462	246	128
	472	924	924	502	924	924	924	924	924	472	472	472	924	924	924	924	924	924	924	472	924	924	924	502	261
	1,1	1,0	1,0	1,1	1,0	1,0	1,0	1,0	1,0	1,1	1,1	1,1	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,1	1,0	1,0	1,0	1,1	2,2
inversion	ODD			EV^2						ODD	ODD	EV^0								ODD				EV^2	EV^{0-2}

	6-Z10(013457)	6-14(013458)	6-Z13(013467)	6-Z24(013468)	6-27(013469)	6-Z19(013478)	6-Z49(013479)	6-Z24(013468)	6-Z28(013569)	6-Z26(013578)	6-34(013579)	6-30(013679)	6-16(014568)	6-31(014579)	6-20(014589)	6-Z50(014679)	6-8(023457)	6-Z39(023458)	6-21(023468)	6-Z45(023469)	6-Z23(023568)	6-33(023579)	6-Z29(023679)	6-32(024579)	6-35(02468T)	
1 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
2 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
3 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
4 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
5 elem.	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
6 elem.	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**50**
7 elem.	6	6	3	6	6	6	3	6	4	3	6	3	6	6	1	3	3	6	6	4	3	6	3	3	1	**236**
8 elem.	21	21	12	21	21	21	12	21	13	12	21	12	21	21	5	12	12	21	21	13	12	21	12	12	4	**846**
9 elem.	56	56	28	56	56	56	28	56	32	28	56	28	56	56	10	28	28	56	56	32	28	56	28	28	7	**2177**
10 elem.	126	126	66	126	126	126	66	126	70	66	126	66	126	126	24	66	66	126	126	70	66	126	66	66	16	**4936**
11 elem.	252	252	126	252	252	252	126	252	136	126	252	126	252	252	42	126	126	252	252	136	126	252	126	126	26	**9752**
12 elem.	462	462	236	462	462	462	236	462	246	236	462	236	462	462	83	236	236	462	462	246	236	462	236	236	50	**17947**
	924	924	472	924	924	924	472	924	502	472	924	472	924	924	166	472	472	924	924	502	472	924	472	472	105	***35944***
	1,0	1,0	1,1	1,0	1,0	1,0	1,1	1,0	1,1	1,1	1,0	2,0	1,0	1,0	3,3	1,1	1,1	1,0	1,0	1,1	1,1	1,0	1,1	1,1	6,6	
			ODD				EV^0		EV^2	EV^0		(T)			ODD	ODD	ODD			EV^2	EV^0		EV^0	EV^0	EV^{0-2}	

Figure 1.34 Enumeration of the 2-object multiset classes. Symmetries: 1,1 EV^0; 1,1 ODD; 2,2 $EV^{0\text{-}2}$.

	Partitions multiplicity of each object	Permutations (# of terms)!	multiplicity of multiplicity	mfunctions (compositions)	# of palindromic compositions	1,1 EV^0 or 1,1 ODD sym.	# with 2 deg. of T-sym (T_6)	# of "circular palindromes"	2,2 $EV^{0\text{-}2}$ symmetry
2 elem.	1+1	2! / 2!		= 1	1	1	1	1	1
					1	**1**	1	1	**1**
3 elem.	2+1	2! / 1! * 1!		= 2	0	1	0	1	1
					0	**1**	0	1	**1**
4 elem.	3+1	2! / 1! * 1!		= 2	0	1	0	1	1
	2+2	2! / 2!		= 1	1	1	1	1	1
					1	**2**	1	2	**2**
5 elem.	4+1	2! / 1! * 1!		= 2	0	1	0	1	1
	3+2	2! / 1! * 1!		= 2	0	1	0	1	1
					0	**2**	0	2	**2**
6 elem.	5+1	2! / 1! * 1!		= 2	0	1	0	1	1
	4+2	2! / 1! * 1!		= 2	0	1	0	1	1
	3+3	2! / 2!		= 1	1	1	1	1	1
					1	**3**	1	3	**3**
7 elem.	6+1	2! / 1! * 1!		= 2	0	1	0	1	1
	5+2	2! / 1! * 1!		= 2	0	1	0	1	1
	4+3	2! / 1! * 1!		= 2	0	1	0	1	1
					0	**3**	0	3	**3**
8 elem.	7+1	2! / 1! * 1!		= 2	0	1	0	1	1
	6+2	2! / 1! * 1!		= 2	0	1	0	1	1
	5+3	2! / 1! * 1!		= 2	0	1	0	1	1
	4+4	2! / 2!		= 1	1	1	1	1	1
					1	**4**	1	4	**4**
9 elem.	8+1	2! / 1! * 1!		= 2	0	1	0	1	1
	7+2	2! / 1! * 1!		= 2	0	1	0	1	1
	6+3	2! / 1! * 1!		= 2	0	1	0	1	1
	5+4	2! / 1! * 1!		= 2	0	1	0	1	1
					0	**4**	0	4	**4**
10 elem.	9+1	2! / 1! * 1!		= 2	0	1	0	1	1
	8+2	2! / 1! * 1!		= 2	0	1	0	1	1
	7+3	2! / 1! * 1!		= 2	0	1	0	1	1
	6+4	2! / 1! * 1!		= 2	0	1	0	1	1
	5+5	2! / 2!		= 1	1	1	1	1	1
					1	**5**	1	5	**5**
11 elem.	10+1	2! / 1! * 1!		= 2	0	1	0	1	1
	9+2	2! / 1! * 1!		= 2	0	1	0	1	1
	8+3	2! / 1! * 1!		= 2	0	1	0	1	1
	7+4	2! / 1! * 1!		= 2	0	1	0	1	1
	6+5	2! / 1! * 1!		= 2	0	1	0	1	1
					0	**5**	0	5	**5**
12 elem.	11+1	2! / 1! * 1!		= 2	0	1	0	1	1
	10+2	2! / 1! * 1!		= 2	0	1	0	1	1
	9+3	2! / 1! * 1!		= 2	0	1	0	1	1
	8+4	2! / 1! * 1!		= 2	0	1	0	1	1
	7+5	2! / 1! * 1!		= 2	0	1	0	1	1
	6+6	2! / 2!		= 1	1	1	1	1	1
					1	**6**	1	5	**6**
			Totals:			**36**			**36**

Figure 1.35 Multiset Class Totals: Dyadic Parent Classes

	2-1(01)	2-2(02)	2-3(03)	2-4(04)	2-5(05)	2-6(06)	
1 elem.	x	x	x	x	x	x	0
2 elem.	**1**	**1**	**1**	**1**	**1**	**1**	**6**
3 elem.	1	1	1	1	1	1	**6**
4 elem.	2	2	2	2	2	2	**12**
5 elem.	2	2	2	2	2	2	**12**
6 elem.	3	3	3	3	3	3	**18**
7 elem.	3	3	3	3	3	3	**18**
8 elem.	4	4	4	4	4	4	**24**
9 elem.	4	4	4	4	4	4	**24**
10 elem.	5	5	5	5	5	5	**30**
11 elem.	5	5	5	5	5	5	**30**
12 elem.	6	6	6	6	6	6	**36**
	36	**36**	**36**	**36**	**36**	**36**	***216***
symmetry	1,1	1,1	1,1	1,1	1,1	2,2	
inversion	ODD	EV^0	ODD	EV^0	ODD	$EV^{0\text{-}2}$	

Figure 1.36 Enumeration of the 10-object multiset classes. Symmetries: 1,1 ODD; 1,1 EV^2; 2,2 $EV^{0\text{-}2}$.

	Partitions multiplicity of each object	Permutations (# of terms)!	/ multiplicity of multiplicity	= mfunctions (compositions)	*# of palindromic compositions*	**1,1 ODD symmetry**	*# of "circular palindromes"*	**1,1 EV^2 symmetry**	*# with 2 deg. of T-sym (T_6)*	**2,2 $EV^{0\text{-}2}$ symmetry**
10 elem.	1+1+1+1+1+1+1+1+1+1	10!	/ 10!	= 1	*1*	1	*1*	1	*1*	1
					1	**1**	*1*	**1**	**1**	**1**
11 elem.	2+1+1+1+1+1+1+1+1+1	10!	/ 9! * 1!	= 10	*0*	5	*2*	6	*0*	3
					0	**5**	*2*	**6**	**0**	**3**
12 elem.	3+1+1+1+1+1+1+1+1+1	10!	/ 9! * 1!	= 10	*0*	5	*2*	6	*0*	3
	2+2+1+1+1+1+1+1+1+1	10!	/ 8! * 2!	= 45	*5*	25	*5*	25	*5*	15
					5	**30**	*7*	**31**	*5*	**18**
			Totals:			**36**		**38**		**22**

Figure 1.37 Multiset Class Totals: Decachordal Parent Classes

	10-1(0123456789)	10-2(0123456781)	10-3(0123456791)	10-4(0123456891)	10-5(0123457891)	10-6(0123467891)	
1 elem.	x	x	x	x	x	x	
2 elem.	x	x	x	x	x	x	
3 elem.	x	x	x	x	x	x	
4 elem.	x	x	x	x	x	x	
5 elem.	x	x	x	x	x	x	
6 elem.	x	x	x	x	x	x	
7 elem.	x	x	x	x	x	x	
8 elem.	x	x	x	x	x	x	
9 elem.	x	x	x	x	x	x	
10 elem.	**1**	**1**	**1**	**1**	**1**	**1**	**6**
11 elem.	5	6	5	6	5	3	**30**
12 elem.	30	31	30	31	30	18	**170**
	36	**38**	**36**	**38**	**36**	**22**	***206***
symmetry	1,1	1,1	1,1	1,1	1,1	2,2	
inversion	ODD	EV^2	ODD	EV^2	ODD	$EV^{0\text{-}2}$	

Figure 1.38 Enumeration of the 1-, 11-, and 12-object multiset classes. Symmetries: 1,1 ODD; 1,1 EV^{2}; 2,2 $EV^{0\text{-}2}$.

	Partitions multiplicity of each object	Permutations (# of terms)!	multiplicity of multiplicity	mfunctions (compositions)	# of palindromic compositions	1,1 EV^{1} symmetry
1 elem.	1	1! / 1!	= 1	*1*	**1**
2 elem.	2	1! / 1!	= 1	*1*	**1**
3 elem.	3	1! / 1!	= 1	*1*	**1**
4 elem.	4	1! / 1!	= 1	*1*	**1**
5 elem.	5	1! / 1!	= 1	*1*	**1**
6 elem.	6	1! / 1!	= 1	*1*	**1**
7 elem.	7	1! / 1!	= 1	*1*	**1**
8 elem.	8	1! / 1!	= 1	*1*	**1**
9 elem.	9	1! / 1!	= 1	*1*	**1**
10 elem.	10	1! / 1!	= 1	*1*	**1**
11 elem.	11	1! / 1!	= 1	*1*	**1**
12 elem.	12	1! / 1!	= 1	*1*	**1**
		Totals:		*12*	**12**

	Partitions multiplicity of each object	Permutations (# of terms)!	multiplicity of multiplicity	mfunctions (compositions)	# of palindromic compositions	1,1 EV^{1} symmetry
11 elem.	1+1+1+1+1+1+1+1+1+1+1	11! / 11!	= 1	*1*	1
					1	**1**
12 elem.	2+1+1+1+1+1+1+1+1+1+1	11! / 10! *1!	= 11	*1*	6
					1	**6**
			Totals:			**7**

Figure 1.38 (continued)

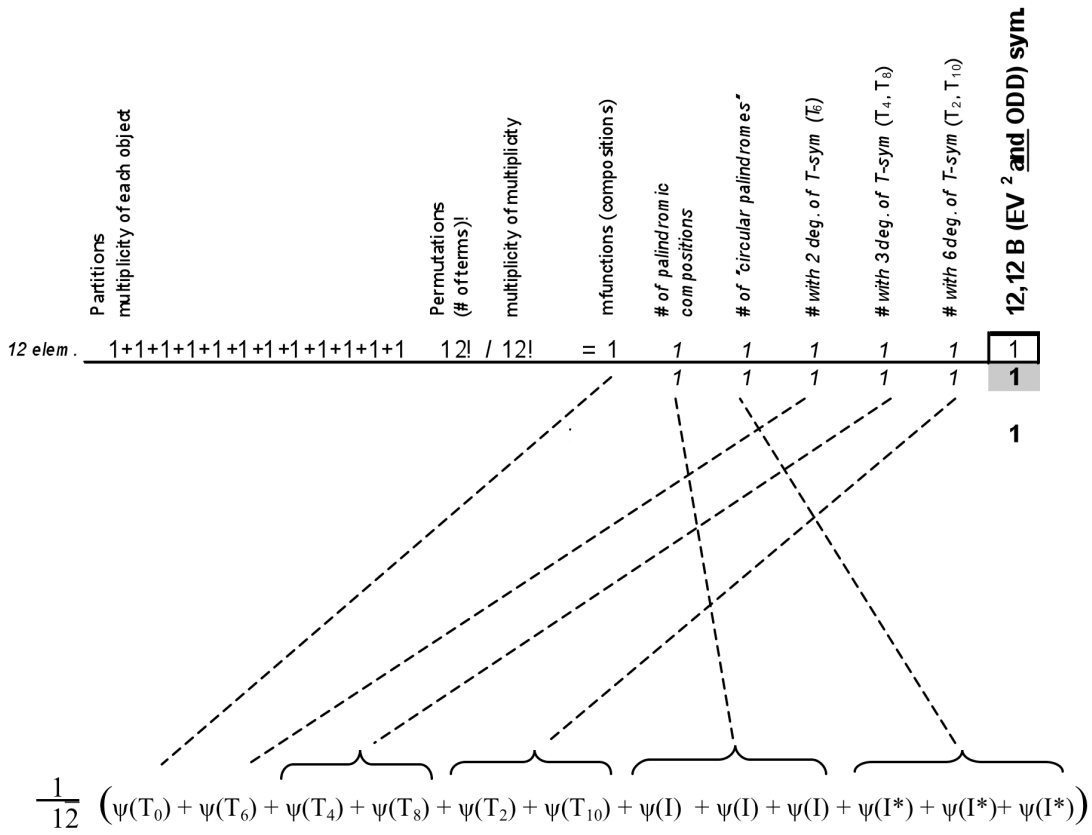


Figure 1.39 Multiset Class Totals: Monadic, Hendecachordal, and Dodecachordal Parent Classes

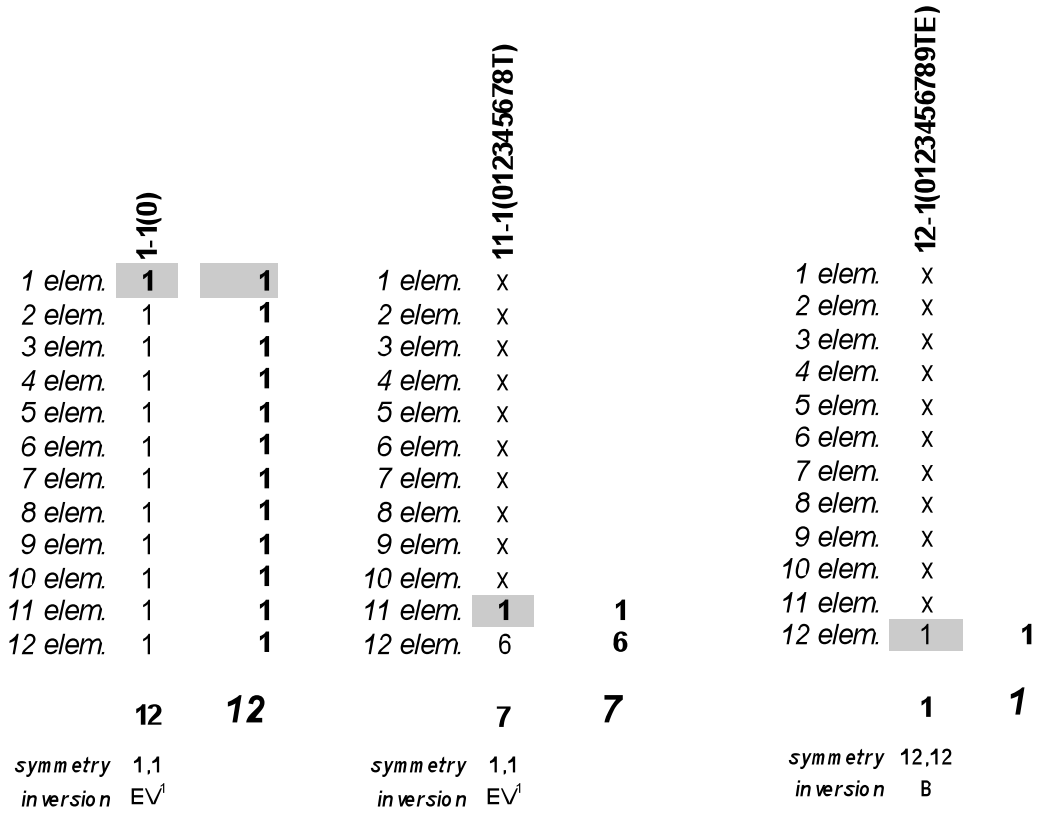


Figure 1.40 Multiset Class Totals: All Parent-Class Cardinalities

	1 ob.	2 ob.	3 ob.	4 ob.	5 ob.	6 ob.	7 ob.	8 ob.	9 ob.	10ob.	11 ob.	12 ob.	
1 elem.	**1**												**1** *monad*
2 elem.	1	**6**											**7** *dyads*
3 elem.	1	6	**12**										**19** *trichords*
4 elem.	1	12	30	**29**									**72** *tetrachords*
5 elem.	1	12	60	85	**38**								**196** *pentachords*
6 elem.	1	18	97	225	170	**50**							**561** *hexachords*
7 elem.	1	18	145	420	510	236	**38**						**1368** *septachords*
8 elem.	1	24	200	754	1170	846	236	**29**					**3260** *octachords*
9 elem.	1	24	267	1170	2340	2177	944	170	**12**				**7105** *nonachords*
10 elem.	1	30	340	1780	4188	4936	2792	780	85	**6**			**14938** *decachords*
11 elem.	1	30	425	2500	6980	9752	6980	2500	425	30	**1**		**29624** *hendecachords*
12 elem.	1	36	517	3469	10940	17947	15296	6913	1526	170	6	**1**	**56822** *dodecachords*
	12	216	2093	10432	26336	35944	26286	10392	2048	206	7	1	***113973***

1.3.4 Object-orientation and Element-orientation

The preceding enumeration of the multiset classes systematically applied mfunctions to the non-multiset parent set classes. The method required the number of positions in the mfunction to equal the cardinality of the parent. It thus positions the canonical set classes as progenitors and the multiset classes as progeny. Offspring not only must contain at least one of each of the parent's pcs, but also must contain *no other* pc. In this view, mscs 02255 and 0025 both are subordinate to sc 3-7(025). This is implied in the preceding taxonomies, and the accompanying harmonic implications should not be overlooked. To subordinate the mset classes to the set classes seems to prefer number of objects over number of elements. This *object-orientation* accounts for multiplicity of its elements but ultimately defers to the parent. It counts pitch-class duplications, but ultimately respects the collection of pitch-classes (not pitch-class representatives) as the "chord" in question. In other words, the object-orientation of multisets views mscs 048, 0048, 00448, and 004488 all as "augmented triads." They may vary in cardinality (number of elements), but these multiset classes all are the same "chord". At a glance, one can see the three distinct objects in play: 0, 4, 8.

This orientation is useful if we prefer to view multisets as *enhancements* of the canonical set classes. For decades, pc duplications at the octave have been routinely ignored during the assessment of a given chord in analysis. We have counted the number of objects while *dis*counting the number of elements. Additional elements, so long as they are not new pitch classes, are seen as incidental. They may contribute to the timbre or texture of the piece, but not to the harmony. This object-orientation of multisets, therefore, respects "traditional" harmonies while still accounting in some way for the

duplications. It is as if each pitch class is given its own volume knob, and one can increase or decrease the relative intensity of each pc.

Another enumeration, a far simpler one to be sure, might take a different approach. An *element-orientation* might universally feature a 12-position mfunction, one that may contain entries of 0. In this orientation, all pcmultisets and multiset classes always contain twelve objects; they simply might contain none of some. Mset class 001447 would appear thus: <2,1,0,0,2,0,0,1,0,0,0,0,0>. Taken for granted is the number of objects; featured is the number of elements. Robert Morris takes exactly this perspective in his enumeration of mosaics in the double aggregate.[51] It is advantageous with respect to certain computations, certainly, but it is also an efficient way to examine voice leading. For example, the pcmultisets W and X, when represented by the following mfunctions: W <0,1,1,1,0,0,0,0,0,0,0,0> and X < 0,1,2,0,0,0,0,0,0,0,0,0>, are seen to be separated by only one semitonal motion. An entry in the 4th position of X moves to the 3rd position resulting in X. The 4th position in X is now zero, but at least there is a zero still there! With regard to voice leading, this move is no different from the following move: Y <0,2,2,2,0,0,0,0,0,0,0,0> to Z <0,2,3,1,0,0,0,0,0,0,0,0>. Here, too, one entry in the 4th place moves to the 3rd place. Let us examine the same two pairs of chords with an object-orientation. It is clear in the first pair (W = 123 and X = 122) that one contains three objects; the other, two. Multiset W belongs to mset class 012, while X belongs to mset class 001. In the second pair (Y = 112233 and Z = 112223) each mset has three objects. We have neither lost nor gained objects in the progression from Y to Z. In the object-orientation, the decrease from two representatives of an object to one

[51] Robert Morris, "Pitch-Class Duplication in Serial Music: Partitions of the Double Aggregate," *Perspectives of New Music* 41/2 (Summer 2003): 96-119.

representative of that object is far less a perturbation of "harmony" than the decrease from one representative of an object to no representatives of that object. The element-orientation would consider each decrease equally, a subtraction of one. An object-orientation counts number of objects first and, with this harmonic starting point, "turns up the volume" on various members by doubling. An element-orientation begins with twelve empty positions or bowls, and pcs are free to move from place to place, their impact determined by steps taken, not by which bowls are filled. One might even say the element-orientation can map a variety of voice-leadings in a twelve-position space while the object-orientation comes closer to a qualitative assessment of particular chords, enabling perhaps similarity relations among them.

This distinction arises here in this section only because the preceding enumeration decidedly came from the object-orientation, but in this dissertation, I will take both perspectives at various times for different reasons. In fact, my default notation (e.g., 3-object pentachord) nicely suggests both orientations simultaneously.

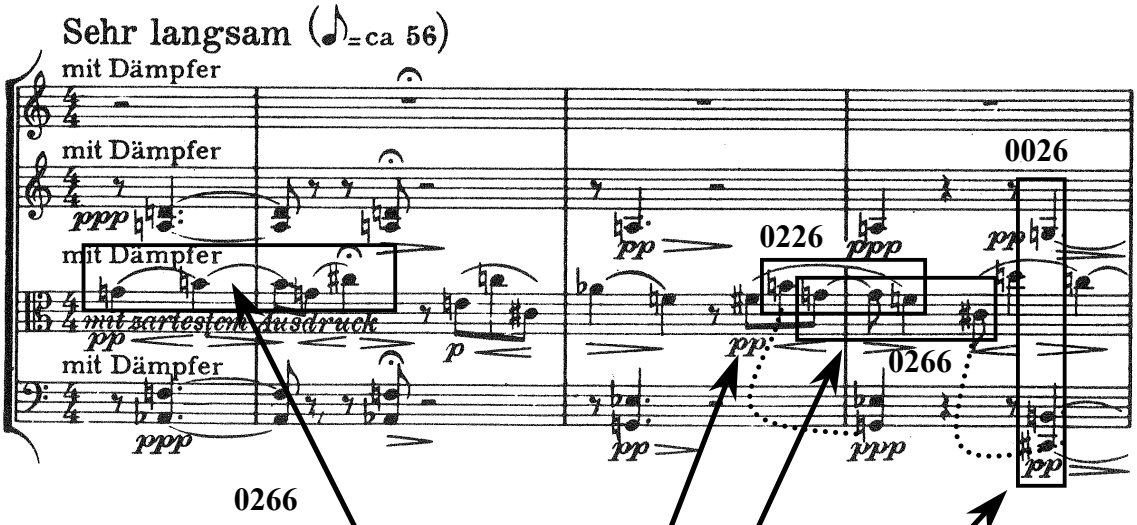
ANALYTICAL INTERLUDE #1

Parent Classes and Multiset Classes in Webern's Opus 5, No. 2

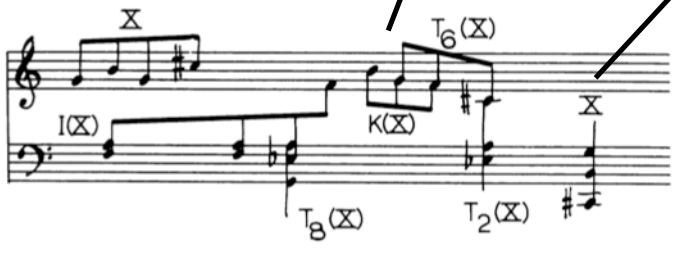
The second of Anton Webern's *Five Movements for String Quartet* is an ideal site for mining some simple pcmultisets and mset classes. To do so is to revisit not only a musical excerpt from the beginning of Chapter 1 but also an article by David Lewin, a analytical essay unarguably seminal in the development of transformational theory.[52] It

Figure 1.41 A parent class and its offspring in Webern, Opus 5, No. 2 (a) Parent class: 3-8 (026); psmsets: 0266, 0226, and 0026. (b) David Lewin's "Example 5" showing transformations of the viola's "*Kopfmotiv*," labeled "X."

(a)



(b)

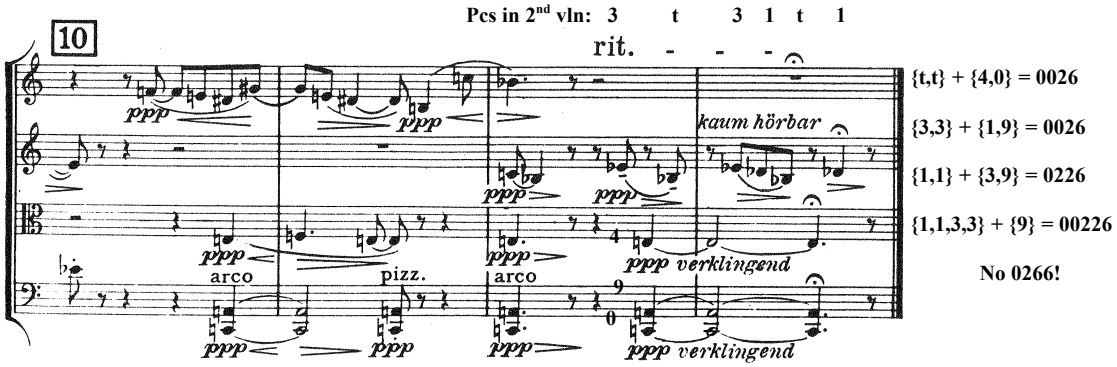


[52] David Lewin, "Transformational Techniques in Atonal and Other Music Theories," *Perspectives of New Music* 21 (1982-3): 312-71.

turns out that pitch-class multisets, which may to some seem strictly to be a computational or set-theoretical enterprise, merge quite nicely with a transformational approach.

Figure 1.41 excerpts the opening of the movement (a) and includes Lewin's relevant transformations of the opening motive in the viola (b). In Lewin's figure the motive is simply labeled X, and its relevant transformed iterations are labeled. ("K" here is a context-specific inversion at index 6.) The arrows linking (a) and (b) point to where the motive X and its transformations are manifested in the score. Each instance of X, whether transposed or inverted, is a member of 3-8 (026), but if each instance of X were to include the nearby doubled member (see dotted lines linking pcs to their doubles in K(X), T_6X, and X), each would represent one of the three distinct tetrachordal offspring of 3-8 (026): 0026, 0226, 0266. The viola's opening motive carries a specific doubling, 0266, but the ensuing music tempts it with alternatives. In the movement's final gesture, at the *ritard*, it is unclear which doubling might be considered primary. The last measure is saturated with motive X, but the motive's original mset class, 0266, is absent. See Figure 1.42. The second violin is given two tokens of pc3, two of pc1, and two of pc10.

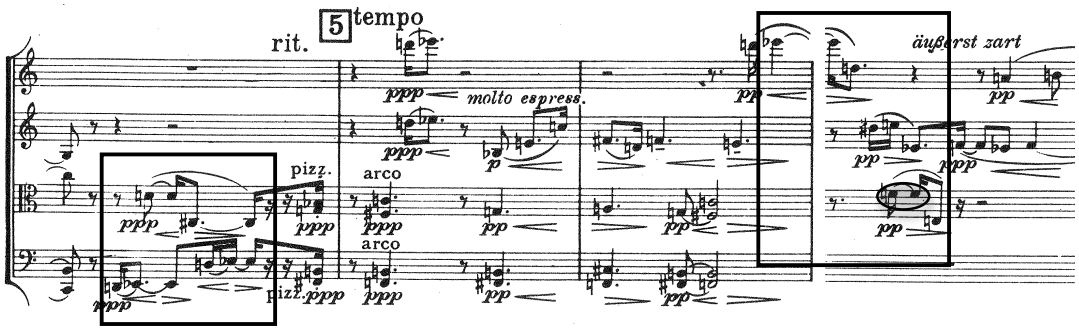
Figure 1.42 Mset classes 0026, 0226 close out Webern, Opus 5, No. 2



Each pair can represent a different emphasis in the off-sprung mset class. The pair of even pcs (pc10) may combine with pcs 0 and 4 to form 0026, while the odd pairs (two pc3s and two pc1s) combine with pc 9 to form either 0026 or 0226 (or even 00226), but the original mset class, 0266, is not feasible.

Measures 4 and 7, each a momentary respite from the sustained three- and four-note chords, contain interlocking voices making great leaps (13 semitones) or small steps (1 or 2 semitones). In Figure 1.43, pcs are collected in each measure, and in each case the resulting collection is a member of 3-1 (012). One might account for these as dense chromatic clusters, but each cluster has the same profile of multiplicity. That is, each collection is a member of mset class 011122. Not only does each collection have the same mfunction, <1,3,2>, but this particular mfunction provides for a unique multiplicity for each member: 1 of one pc, 2 of another, and 3 of another. It gives the cluster a distinctive shape: fat in the middle and leaning to one side. This can be heard in measure 4 as pc2 is thrice iterated, each time shifting up or down a semitone. Similarly, in measure 7, pc3 is the center of the cluster pivoting more to pc4 than to pc2. (The pc5 in

Figure 1.43 Mset class 011122 appears twice in Webern, Opus 5, No. 2



{1,2,2,2,3} = 01112 {2,3,3,3,4} = 01112

the viola is overlooked and grayed out here, but does, with the following pcs, create more offspring of 3-1(012).) A similarity like this frequently is an almost inevitable result of invertible, imitative counterpoint, to be sure, but assessing the doubling in a chord like this gives extra dimension to an otherwise flat chromatic cluster.

CHAPTER TWO: IMPLICATIONS

2.1 The Interval-Class Vector and Common Bonds

The interval-class vector forms the basis for many other concepts in atonal theory. Notions of similarity, complementation, and even chord quality make use of it. That is to say, in addition to concisely reporting the intervallic content of a single pitch-class collection, the interval-class vector also serves reliably to compare several.[1] The common tone is a concept with a long history, but in pcset theory, the term usually refers to a pitch class that is an element in two or more pc collections. Because no pcset may contain more than one instance of a pc, a common tone (common pc) between two pcsets appears only once in each. Between pcmultisets, however, a common tone (common pc representative) may appear more than once in either pc multiset. In such a scenario, there is still only one common pitch class, but the number of common representatives potentially increases. This leads to an important question: how many elements do the following two pcmsets hold in common: {2, 3, 4, 4} and {4, 5, 6, 7}? There clearly is one *object* in common—pc 4—but it is not clear how many *elements* are shared. Is it two (the first holds two in common with the second) or one (the second holds one in common with the first)? Should one cover all bases and say there are three like elements (representatives of pc4) altogether?

This section discusses interval-class vectors for pcmultisets and also proposes a solution to the common-tone problem outlined above. Owing to their relatedness—the

[1] See Allen Forte's Interval Vector in *The Structure of Atonal Music* (New Haven: Yale University Press, 1973): 15-18.

vector can predict common tones under transposition—the two topics are presented in the same section here.

2.1.1 The Interval-Class Vector

The primary function of the interval-class vector (ICV) is to report the multiplicity of each interval class in a given pitch-class set (or set class). In its 6-position form, the ICV tallies only the interval classes (ics) 1–6, returning six values. The 7-position ICV returns an additional value; it tallies the number of intervals belonging to ic0. Because a pitch-class set can have only one instance of any single pitch class—the pitch-class integer stands in for all of its potential representatives—the only unisons that will be tallied in a 7-position vector for pcsets are those intervals measured from a pitch class to itself. Such ic0s are as valid as any of the other six interval classes; the pcset {0, 4, 7} might be represented by $C_4 – E_4 – G_4 – C_5$, and the ic0 formed by the C doubled at the octave should be reported as well as the other ics, 3, 4, and 5. In practice, however, this position and its tally are usually discarded. For one thing, it simply and predictably reports the cardinality of the pcset. For another, only when a pc is measured with itself does an ic0 result. The other ics, however, require the interaction of pairs of different pcs. Therefore the multiplicity of these six ics can vary dramatically, depending on the combination of pcs present, while the number of ic0s will be predictable and usually will be considered trivial.

Pitch-class multisets, however, are a different matter. As pcmultisets account for doubling of pcs, so must their interval-class vectors account for ic0s. A pcmultiset's ICV, then, will have seven positions. The unison position, however, must *not* report the

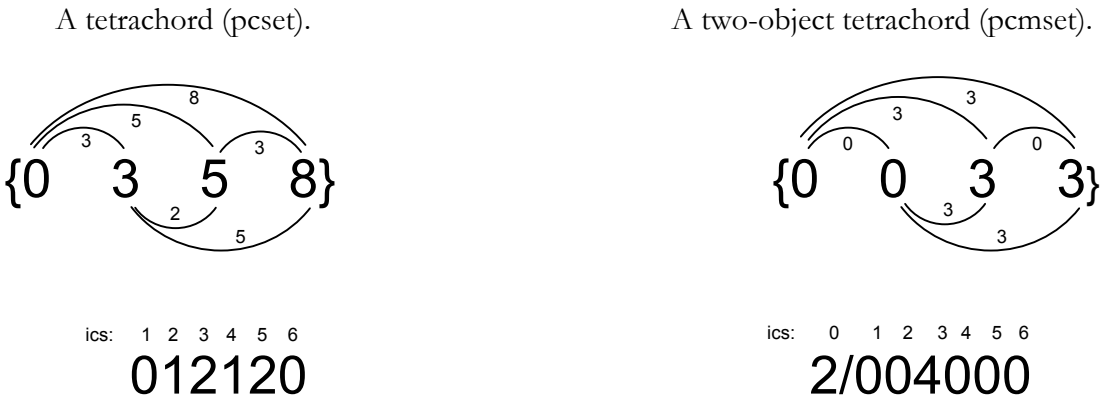
ic0 measured from an element to itself. This is so because the elements are not pitch classes but *representatives* of pitch classes. Let us recall the following definition from Chapter One: *A pitch-class multiset is an unordered collection of pitch classes (not necessarily distinct), in which each element denotes a single representative of its pitch-class.* Therefore, the ICV of the *pc*set {0,3,7} rightly would report three unisons, but the ICV of the pc*multi*set {0,0,3,7} would not. Nor would it even report four. Since each of the four elements denotes a *single* representative of each pc, one should tally only the interval classes measured from one element to another, not from an element to itself. As a result, there would be only one instance of ic0. In the pcmset {0,0,3,7} there are four ptokens, and two of them simply belong to pitch class 0. The lone ic0 measured between the two should be tallied just as the ic between any other two pc representatives, thus producing the following vector: 1/002120.[2] The slash, separating the ic0 column from the remaining six, is in place only to assist readers who are more familiar with the 6-position vector.

Figure 2.1 compares the tallying process in a pcset with that in a pcmset. The interval-class tally is performed on pcset {0,3,5,8} in the traditional manner, resulting in the vector 012120. To the right is an interval-class tally of the pcmultiset {0,0,3,3} resulting in the vector 2/004000. There are two unisons and four ic3s. The procedure for both tallies is identical, as seen in the number of intervals (here represented by arcs), so in each case we are left with a total of six vector entries. In this respect, the new vector is

[2] One could simply add 4 to the tally of ic0s in this example in order to reflect the self-mapping of each element, but I prefer to emphasize the discreteness of each pc representative and count only the intervals *between* them. Only later, when comparing a pcmultiset with a T_n of itself, will I account for this self-mapping. Until that point, as a simple indicator of intervals present within a pcmultiset, such ic0s are not considered.

fully compatible with the six-position one.[3] Furthermore, any pcmultiset whose number of elements and number objects are the same will have the same ICV pcset parent, but with a zero added in the additional, first position.

Figure 2.1 Calculation of the interval-class vector in pcsets and pcmultisets



2.1.2 Common Bonds

Identifying common tones (pcs) in a pair of pcsets is a simple task. One simply counts the number of pcs that appear in both. Since a pc is never duplicated in a pcset, any pc held in common appears once and only once in each of the pc sets. Strictly speaking, the same is true for common pcs in multisets. For example, the following pcmsets have one pc in common: X={2,4,4,4,7} and Y={4,5,5}. Pitch class 4 appears in both. The pcmset X has three representatives while pcmset Y has only one. However, since "common tones" among pcsets are understood as common pcs, "common tones" among pcmsets should be understood as common pc representatives. How many

[3] Again, the number of elements could be added to the ic0 column, and the pcmset's ICV would then be compatible with the pcset's 7-position one. Because the 6-position vector is vastly more prevalent in the literature than the 7-position one, however, it is more desirable to match total multiplicity in the former than in the latter. Counting inter-representative but not intra-representative ic0s allows this.

common pc representatives are there, then, between X and Y? With respect to Y, pcmset X has three pc representatives in common, but with respect to X, pcmset Y has only one pc representative in common. Which is the preferred answer? Because common tone mappings between pcmultisets are likely not to be one-to-one, the common tone is discarded in favor of the **common bond**.

Figures 2.2 and 2.3 show the common tone and the common bond respectively. The pcsets in Figure 2.2 share one pc in common, and the one-to-one correspondence of this pc is clearly seen. The pcmsets in Figure 2.3 differ in their number of representatives of the shared pc at a ratio of 3-to-1. Each of the three lines here represents a common bond. This bond is not necessarily a mapping, but simply a way to measure the relative prevalence of a particular pc in multiple pcmsets. For any two pcmsets, the total number of common bonds will be the sum of the products of the number of each pc representative in each pcmset. Figure 2.4 demonstrates with two large pcmsets.

Figure 2.2 One common <u>tone</u> between pcsets

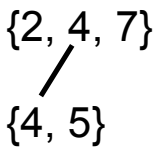


Figure 2.3 Three common <u>bonds</u> between pcmsets

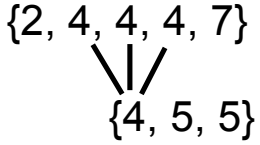


Figure 2.4 Total number of common bonds as the sum of the product of the multiplicities of pc representatives in two pcmsets. (# = "multiplicity of")

pcmset X	**pcmset Y**	
{1, 1, 3, 3, 4, 7, 7, 7}	**{0, 1, 2, 2, 5, 5, 7, 7, 9, T}**	
#0 = 0	#0 = 1	0*1 = 0
#1 = 2	#1 = 1	2*1 = 2
#2 = 0	#2 = 2	0*2 = 0
#3 = 2	#3 = 0	2*0 = 0
#4 = 1	#4 = 0	1*0 = 0
#5 = 0	#5 = 2	0*2 = 0
#6 = 0	#6 = 0	0*0 = 0
#7 = 3	#7 = 2	3*2 = 6
#8 = 0	#8 = 0	0*0 = 0
#9 = 0	#9 = 1	0*1 = 0
#T = 0	#T = 1	0*1 = 0
#E = 0	#E = 0	0*0 = 0
		sum =8 common bonds

2.1.3 Common Bonds Under Transposition

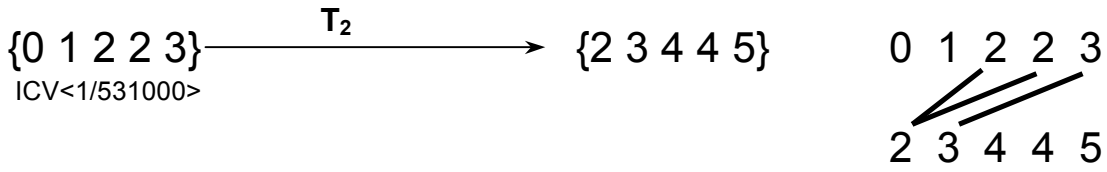
In addition to being a means for tabulating interval classes in a pcset, the interval-class vector is also a predictor of multiplicity of common tones under transposition; every interval-class x between elements in a pcset represents a potential mapping of one pitch class to the other (i.e., a tone held in common) under T_x.[4] Every interval-class 6 represents, however, a two-way mapping of its two pitch classes (i.e., two tones held in common). In largely the same way—there is a twist—the seven-position vector predicts the number of common bonds between a pcmultiset and its transposition.

Figure 2.5 demonstrates. At left is the pcmultiset {0, 1, 2, 2, 3}, a four-object pentachord. Its ICV is 1/531000. The entries add up to ten, as we would expect to be the

[4] See Milton Babbitt, "The Structure and Function of Musical Theory," *College Music Symposium* 5 (Fall, 1965): 49-60, reprinted in *The Collected Essays of Milton Babbitt*, ed. Stephen Peles, Stephen Dembski, Andrew Mead, Joseph N. Straus (Princeton, NJ: Princeton University Press, 2003), 191-201. Babbitt not only offers a crisp explanation of the principle, but also uses it to make the case for the structural fruitfulness of both the 12-tone system and the diatonic scale. See also Forte, *Structure*, pp. 34-35.

case in the tally of intervals in any pentachord, and there is a 3 in the ic2 position. This predicts three common bonds under T_2. This transposition yields pcmultiset {2, 3, 4, 4, 5}. On the right-hand side of the figure, the original and the transposition are compared. There are, as anticipated, three common bonds displayed. This procedure will be the same for any T_x when $x \neq 0$, except when x=6, the number of common bonds is twice the number appearing in the ic6 position of the vector. (This is identical to the standard procedure for common tones in pcsets.) Because these bonds are not always one-to-one but often two-to-one or greater, however, they should be understood as potential mappings if mappings at all. It is the potential mappings of the transformation that predict the number of bonds, which measure, primarily, similarity of relative multiplicity in the results.

Figure 2.5 The ICV as predictor of number of common bonds under T_x ($x \neq 0$).

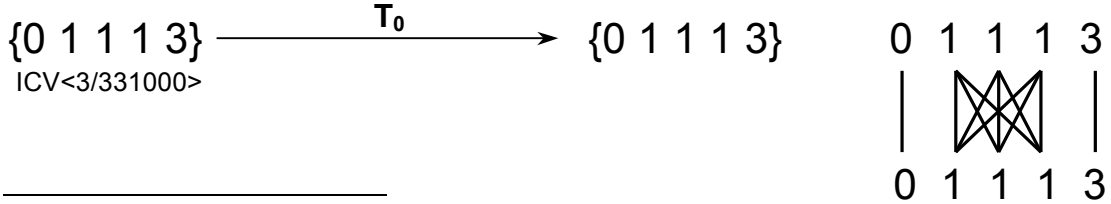


An additional, yet perfectly reasonable, step must now be taken in the case of T_0. Earlier, when calculating the 7-position ICV for multisets, I excluded for practical reasons the self-mapping ic0s, and I counted only those ic0s between repeated pc representatives in the pcmset. When considering potential common bonds between pcmsets, this self-mapping ic0 tally (equivalent to the cardinality of the pcmset) must be reinstated in addition to the ic0s between duplicate representatives but only after the original ic0 tally is doubled. For example, pcmset {5a, 7, 0, 5b} contains two representatives of pc5. At T_0 ={5a′, 7′, 0′, 5b′} these representatives can map onto each

other, forming two different common bonds: 5a′–5b and 5b–5a′. So while the ICV will show only one ic0, that "1" represents two common bonds at T_0. In this respect, the ic0 tally is treated like the ic6; it is doubled. Finally, we add the cardinality of the set. This accounts for the unisons between the elements in one set and the same elements (in the same positions) in the other set. Further below, section 2.1.6 shows why these steps (for ic0 and ic6) are necessary when using the ICV and proposes an alternative already in use.

Figure 2.6 clarifies, showing both types of common bonds. Here, pcmultiset {0, 1, 1, 1, 3} is transposed by T_0 to produce {0, 1, 1, 1, 3}. The ICV contains only 3 in the ic0 position, but a web of many more common bonds appears at the right, between original and transposition. Here is how 3 becomes 11: the original 3 value gets doubled to six, and these six are represented by the six diagonal bonds in the figure, connecting pc representatives in one pcmultiset to pc representatives of the same object (but in different positions) in the other pcmultiset. The cardinality (the self-mappings) gives us five which is reflected in the five vertical bonds in the figure. Thus, we have a total of eleven common bonds.[5]

Figure 2.6 The ICV as predictor of number of common bonds under T_0. (3*2 + card. = 11 common bonds)



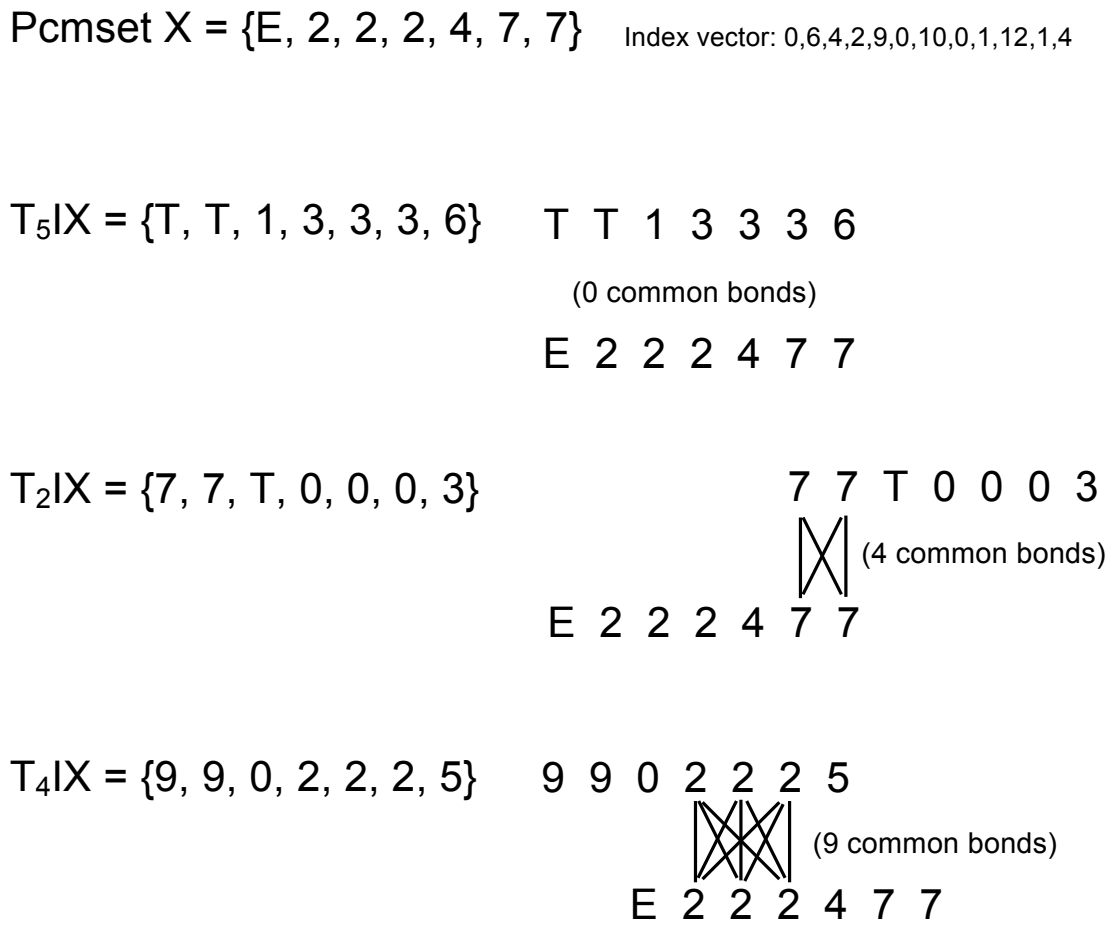
[5] These vertical bonds in the figure are similar to but not the same as the identity function I prohibited earlier. In this case we have two iterations of the pcmultiset: one original and one at T_0. The 0 in one pcmset is as independent of the 0 in another pcmset as it is of a different 0 in its own. What I did not include in the ICV was mapping of a pc representative to itself before any operation was even enacted upon it. An element in a pcset is a pitch class representing many members and can by virtue of its identity, "self-map" without any operation being acted upon it. An element in a pcmultiset is a single representative of the designated pitch class and requires an operation before it can map to another element.

2.1.4 Common Bonds Under Inversion

Finding common tones under inversion between pcmultisets requires even less revision of the standard tools. The index vector operates correctly, in the standard fashion, so long as distinct representatives of a pc are treated as individual elements.

Figure 2.7 shows pcmultiset X={E, 2, 2, 2, 4, 7, 7, 11} and its index vector.[6] The

Figure 2.7 The index vector of a pcmultiset predicting common bonds under inversion.



[6] See John Rahn's TICS in *Basic Atonal Theory* (New York: Longman, 1980), 111-12 and Joseph N. Straus's index vector in *Introduction,* 84, both of which track common tones under inversion. Each of the twelve positions in this vector first lists the number of pairs of pc representatives in the pcmsest that sum to a particular index number, 0-11. Since each single entry represents a potential mapping of *two* pc representatives onto one another at that index of inversion (i.e., two mappings), the procedure then requires doubling of each entry. The last step is to consider self-mapping of each pc representative. Each pc representative is finally summed with itself, constituting the remaining index entries.

Vector's positions represent indexes of inversion, 0-11.[7] The number in each entry represents the number of pc representatives held in common under inversion at that particular index. Below the pcmultiset in the same figure are three different inversions of pcmset X: T_5I, T_2I, and T_4I. To the right of each inversion is a side-by-side comparison of the orginal pcmset and its inversion. The common bonds are drawn in, and in each case the number of common bonds correctly matches the entry in the index vector for the particular inversion in question. Once again, the nine common bonds between X and T_4IX are not *the* mappings of this transformation. After all, the three pc2s in X can map onto the pc2s in T_4IX in only six possible ways. These bonds do, however, represent *all possible* individual mappings at once. It is a post- or pre-transformational assessment of the relation between the two pcmsets.

2.1.5 Difference and Sum Matrices

It should not be surprising that current tools work well for pcmultisets. When additional representatives of a pc are treated the same as any other entry, the tabulation of intervals and indexes proceeds without error. The additional step incurred in 2.1.4, as minor as it is, is a complication due neither to pcmultisets nor to admitting ic0 in the vector. This is because the ICV, which deals in interval classes (*unordered* intervals), is being used to predict results from transposition, an operation which deals in *ordered* intervals. This is old news, of course, and anyone using the ICV in such a way is accustomed to doubling the value in the ic6 position of the vector. Now that the ic0 column similarly must be doubled (before considering self-mappings), adding another

[7] Commas are necessarily added here because while the number of entries per index is limited when working with pcsets, the sky is the limit with pcmsets because the pc doubling is limitless.

step, it might be better to consider the difference matrix for this purpose. The sum matrix is already prevalent in the literature and is in common in the classroom during the teaching of inversion.

Figure 2.8 shows a difference matrix applied to the pcmset {3, 5, 5, 9}.[8] The pcsmet is repeated across the top and down the left. The entries in the matrix are calculated by subtracting the pc representatives on the left from those on the top, mod 12. These entries, then, represent potential transposition levels from one pc representative to another. The ICV for {3, 5, 5, 9} is 1/020201. Looking only in the lower-left (or upper-right) half of the matrix, not including the diagonal itself, one can see an immediate correlation to the ICV. There are two ic2s, two ic4s, an ic0, and an ic6. When we double the entries in the ic0 and ic6 column to consider potential transpositions, we are simply accounting for the reappearance of those ics on the other diagonal half of the matrix. The number of ics 1-5 do not need to be doubled in the same way because their mirror-image appearances across the diagonal are their *ordered* interval complements, the same interval class, but representing a different potential transposition level. The diagonal here represents the four unisons that are added in the procedure.[9]

The sum matrix is commonly invoked in discussions of pcset inversion, its advantage being its ability to predict not only how many common tones will be held in common under various indexes, but also exactly *which* pcs will be held in common. Using it with pcmultisets poses no problems and no ambiguities, as Figure 2.9 shows.

---

[8] Carlton Gamer and Paul Lansky demonstrate the difference matrix applied to two *different* pcsets as well as to the same one in "Fanfare for the Common Tone," *Perspectives of New Music* 14/2 (1976), 229-235.
[9] These four unisons could, of course, already be in the ICV if self-mappings were counted from the start.

Figure 2.8 The difference matrix applied to pcmultiset {3, 5, 5, 9}. Values in the matrix are values of T_n, potential transpositions producing common bonds.

T_n	3	5	5	9
3	0	10	10	6
5	2	0	0	8
5	2	0	0	8
9	6	4	4	0

Figure 2.9 The sum matrix applied to pcmultiset {3, 5, 5, 9}. Values in the matrix are values of T_nI, potential inversions producing common bonds.

T_nI	3	5	5	9
3	6	8	8	0
5	8	10	10	2
5	8	10	10	2
9	0	2	2	6

2.2 Pcmultisets and the Z-, Zo-, Ze-, and Zoe-relations

Two set classes are said to be Z-related if they share the same interval-class vector (ICV) but are neither T_n- nor T_nI-related to one another.[10] A relatively rare phenomenon, the Z-relation suggests a peculiar kinship or hidden relationship between two set classes. One might say that with identical interval content, a set class and its Z-correspondent share a certain sonority or "chord quality." That the Z-relation does not have a thriving life in the theoretical literature is a product either of the limited number of set classes it engages or of the arguable perceptibility of any implied similarity. It concerns only 19 pairs (23, including cardinality >6), most of which are hexachord complements, and it cannot be heard anyway. So goes the argument. With respect to set classes (in 12-tone equal temperament) this may be true. When considering doubled pc representatives in multiset classes and using the ICV as described in the previous section, however, the Z-relation proliferates quickly. What was formerly an either/or proposition—scs either are Z-related or are not—becomes a fuzzier one. Two set classes can become Z-related with just the right amount of doubling. To put it differently, two set classes have a degree of Z-relatedness, measured by the amount of pc multiplicity required to achieve the relation.

David Lewin recognized that in 12-tone equal temperament, Z-related set classes always come in pairs, but in the "16-tone scale" they come in triples.[11] In these cases, three unique set classes share the same vector. My study of multisets does not stray from

---

[10] Howard Hanson's 1960 treatise was perhaps the first published discussion of this property: "There are a few sonorities which *have the same components but which are not involutions one of the other*. […] We shall describe such sonorities … as *isomeric* sonorities." (*Harmonic Materials of Modern Music: Resources of the Tempered Scale* (New York: Appleton-Century-Crofts, 1960), 22, italics in original.) Allen Forte introduced and coined the term itself in *The Structure of Atonal Music.* Robert Morris, in "Set Groups, Complementation, and Mappings among Pitch-Class Sets," *Journal of Music Theory* 26/1 (Spring, 1982): 101-144, discusses its relationship with both the hexachord theorem and his ZC-relation.

[11] David Lewin, "On Extended Z-Triples," *Theory and Practice* 7/1 (August, 1982): 38-39.

the 12-tone octave, but it unquestionably allows for the possibility of triples, quadruples, etc. Three, four, or even more distinct multiset classes may, with the right doublings, share the same ICV.

In this section, I examine hundreds of thousands of pcmultisets, seeking the Z-relation among them. In addition I propose three other similar relations: the Zo-, Ze-, and Zoe-relations. The "o" simply indicates a Z-relation between multiset classes that have a different number of objects (pcs), and the "e" indicates a modified Z-relation where the 0-column in the ICV is ignored.

Ultimately, and somewhat ironically, this section is an assessment of set classes, not multiset classes; the canonical set classes (parent class) are related only at the degree to which doubling is necessary to "Z-relate" their multiset offspring.

2.2.1 Introducing the Four Z-type Relations

Because doubling a pc representative in a pcmultiset not only increases pcmultiplicity but also increases multiplicity of the intervals involving that pc, a single parent set class can produce an astonishing variety of interval-class vectors depending on which pcs increase in multiplicity. Amid such variety, the Z-relation flourishes to various degrees. Some pairs of pcsets that are *not* Z-related can be understood as Z-related pcmultisets simply by doubling only a single pc in each pcset.[12] Other pairs are more resistant. Sometimes one must add a dozen or more doublings to each pcset in a pair in order to yield Z-related results. The Z-relation emerges, then, among set classes—more accurately, among the mset class offspring of set classes—that are not themselves

[12] Msetclasses 01245 <1,1,1,1,2> (=012455) and 02347 <1,1,2,1,1> (=023347) share the same ICV: 1/423320

so related. What is more, because two pcmultisets with the same number of elements—a prerequisite for the Z-relation—can come from parent classes of different cardinality, the Z-relation can exist between, say, 3-7(025) and 4-23(0257), so long as one compares the multiset classes produced by the right doubling.[13] This relation, here called a **Zo-relation**, *is* a Z-relation, but the "o" is a simple reminder that the multiset classes in question have a different number of objects, i.e., they come from parent classes of different cardinality.

I make one slight modification to the Z-relation in order to produce the remaining two relations, **Ze** and **Zoe**. If two distinct multiset classes have identical ICVs *except* for the entry in the unison column, they are said to be Ze-related (pronounced "Zay"). The "e" reminds us that the multisets have a different number of elements, which is necessarily so, given the conditions above. What this does is to account for intervals *between* pc representatives (even multiple ones), while dismissing the unisons (octaves) created by the doublings themselves. Doubling a pc, from this perspective, magnifies the intervallic relationships with other pcs but ignores intensification of the single pc. This relation allows for some surprising pairings. Finally, the Zoe-relation is found between Ze-related multiset classes who differ in their number of objects as well.

2.2.2 The Interval-Content Profile

Before beginning an enumeration of Z-, Zo-, Ze-, and Zo-e related pcmultisets, I classified each of the parent classes according to its **interval-content profile**. This profile is a generalized version of the ICV, displaying either an x or a 0 in each of its six

[13] Msetclasses 025 <5,4,3> (=000002222333) and 0257 <3,6,2,1> (=000222222557) share the same ICV: 19/0, 20, 12, 0, 15, 0.

positions. For example sc 4-7(0145) has the following ICV: 201210. Leaving the zeros alone and replacing the other entries with an x produces the following IC profile: x0xxx0. Its purpose here is to sort each of the parent classes by profile, because if two parent classes have different profiles, none of the off-sprung pcmultisets can be Z-related. Nor can they be Zo-, Ze, or Zo-e related. The parent class must have the interval to begin with before its multiplicity can be increased. Searching for related pairs across profiles is thus futile. For example, suppose pcmset A contains an ic5 and pcmset B does not. No amount of doubling of pcs in pcmset A will remove the presence of ic5. Its multiplicity may increase, but the ic5 position in the vector always will read 1 or greater. Likewise, no amount of doubling of pcs in pcmset B will produce an ic5. Owing to their permanent incompatibility with respect to ic5, neither parent can produce a pcmset Z-related to a pcmset from the other.

This profile serves all four relations (Z, Zo, Ze, and Zoe) because it does not include the ic0 column of the IC vector. The Ze- and Zoe-relations disregard this column entirely but still require agreement in the standard six ICV positions.

The IC profile is especially useful when working with set classes containing three or four objects (the canonical trichords and tetrachords). No two trichord parent classes have the same profile, and nearly half of the tetrachord parents have profiles unique for their cardinality. At the other end of the spectrum there is the profile xxxxxx. Representing parent classes with at least one of each interval class, this profile is held by a great many pentachords, most hexachords, and all parent classes of higher cardinality. To manage this disparity more efficiently I grouped the profiles into four profile groups according to the number of parent classes that share each profile. Figure 2.10 lists,

according to profile group number and then by IC profile, all of the parent classes. The table tantalizingly suggests that any two parent classes with same profile can produce Z-related offspring, simply by acquiring the right amount of doubling. Furthermore, one can easily find Z-related triples, quadruples, and so forth. When several of the parent classes produce offspring all with the same ICV. This study neither proves nor disproves the former suggestion, but it discusses to some degree Z-triples.

The table in Figure 2.10 is used for reference both in the following section, which describes the computer programs used for discovering related pairs of pcmsets, and in later sections, which detail the results produced by these programs. These groups are an organizing influence on the piles of data to come.

2.2.3 Computer-Assisted Discovery of Z- and Zo-related Pairs

The procedure here for finding Z- and Zo-related pairs is one of brute force. With the assistance of the computer, I compared the seven-position IC vectors of millions of pairs of pcmultisets. The several programs written for the task did the following: (1) produced for each parent class a list of all off-sprung pcmsets containing up to 25 elements; (2) produced an IC vector for each of these pcmset; and (3) compared these IC vectors to one another where the parent classes fell into the same profile group.

Figure 2.10 Interval-Content profile groups 1-4. An x represents an entry of 1 or greater in the ICV, and a 0 represents an entry of 0 in the ICV.

PROFILE GROUP 1

xxx0xx	**xxx0x0**	**xx00xx**	**xx0000**	**x0xx00**	**x00xxx**	**x00xx0**	**x00000**
4-13 (0136)	4-10 (0235)	4-6 (0127)	3-1 (012)	3-3 (014)	4-8 (0156)	3-4 (015)	2-1 (01)
0x0x00	**0x00x0**	**0x0000**	**00xxx0**	**00x000**	**0000x0**	**00000x**	
3-6 (024)	3-9 (027)	2-2 (02)	3-11 (037)	2-3 (03)	2-5 (05)	2-6 (06)	

PROFILE GROUP 4

xxxxxx
4-Z29 (0137)
4-Z15 (0146)
5-4 (01236)
5-5 (01237)
5-9 (01246)
5-Z36 01247)
5-13 (01248)
5-6 (01256)
5-14 (01257)
5-Z38 (01258)
5-10 (01346)
5-16 (01347)
5-Z12 (01356)
5-24 (01357)
5-19 (01367)
5-29 (01368)
5-31 (01369)
5-Z18 (01457)
5-30 (01468)
5-32 (01469)
5-20 (01568)
5-25 (02358)
5-28 (02368)
5-26 (02458)
6-2 (012346)
6-Z36 (012347)
6-Z37 (012348)
6-9 (012357)
6-Z40 (012358)
6-5 (012367)
6-Z41 (012368)
6-Z42 (012369)
6-Z38 (012378)
6-15 (012458)
6-22 (012468)
6-Z46 (012469)
6-Z17 (012478)
6-Z47 (012479)
6-Z44 (012569)
6-18 (012578)
6-Z48 (012579)
6-Z10 (013457)
6-27 (013469)
6-Z49 (013479)
6-34 (013579)
6-30 (013679)
6-Z29 (023679)
6-16 (014568)
6-31 (014579)
6-21 (023468)
6-Z45 (023469)
6-33 (023579)
6-Z3 (012356)
6-Z4 (012456)
6-Z11 (012457)
6-Z12 (012467)
6-Z13 (013467)
6-Z6 (012567)
6-Z24 (013468)
6-Z43 (012568)
6-Z25 (013568)
6-Z19 (013478)
6-Z26 (013578)
6-Z39 (023458)
6-Z28 (013569)
6-Z50 (014679)
6-Z23 (023568)
ALL SEPTAS
ALL OCTAS
ALL NONAS
ALL DECAS
ALL HENDECS
ALL DODECS

PROFILE GROUP 2

xxxx0x	**xxx000**	**x0xxxx**	**x000xx**	**0xxxxx**
4-12 (0236)	3-2 (013)	4-18 (0147)	3-5 (016)	4-27 (0258)
5-8 (02346)	4-1 (0123)	5-22 (01478)	4-9 (0167)	5-34 (02469)
0xx0x0	**00x00x**	**000x00**	**xxxx00**	**0xxxx0**
3-7 (025)	3-10 (036)	2-4 (04)	4-2 (0124)	4-22 (0247)
4-23 (0257)	4-28 (0369)	3-12 (048)	4-3 (0134)	4-26 (0358)
			5-1 (01234)	5-35 (02479)

PROFILE GROUP 3

xx0xxx	**x0xxx0**	**0x0x0x**	**xxxxx0**
4-5 (0126)	4-7 (0145)	3-8 (026)	4-4 (0125)
4-16 (0157)	4-19 (0148)	4-21 (0246)	4-11 (0135)
5-7 (01267)	4-20 (0158)	4-24 (0248)	4-14 (0237)
5-15 (01268)	4-17 (0347)	4-25 (0268)	5-2 (01235)
6-7 (012678)	5-21 (01458)	5-33 (02468)	5-3 (01245)
	6-20 (014589)	6-35 (02468T)	5-Z17 (01348)
			5-27 (01358)
			5-11 (02347)
			5-23 (02357)
			5-Z37 (03458)
			6-1 (012345)
			6-14 (013458)
			6-8 (023457)
			6-Z32 (024579)

The programs serving function (3) above were designed to compare the pcmsets with the fewest number of elements first, pausing at the first successful match in IC vector. This allowed me to record the pairs pcmsets with the fewest elements, disregarding the infinite number of pairs that may be produced by perpetual doubling. For example, scs 4-5(0126) and 4-16(0157) have the following Z-related offspring: 01266 and 01157. These pcmsets share the vector 1/210222. Once this pair is discovered, the program could continue to find more Z-related pairs with higher multiplicity, but I stopped the program there and recorded the pair with the lowest multiplicity (i.e., the fewest number of elements).[14]

I took note not only of the related pcmsets and their shared IC vectors but also of the number of elements, which reflects the amount of multiplicity. Of all the possible Z-related pairs of pcmsets off-sprung from a pair of parents, I recorded the pair(s) with the fewest elements. This way the many pairs of parent classes, themselves, can be compared with respect to the multiplicity of their "smallest" Z-related offspring. For example, pcmsets 01266 and 01157 are Z-related (ICV=1/210222). To produce this Z-related pair one needs only to add one element (one doubling) to each of the parent scs, 4-5(0126) and 4-16(0157). The parent classes 4-2(0124) and 4-3(0134) also have Z-related offspring, but one would have to add a minimum of eight elements to find them: 011112224444 and 001113333344 (ICV = 15/16, 15, 15, 4, 0, 0).[15] The former pair of

[14] Actually, I continued the program long enough to exhaust possibilities with the same number of elements. Sometimes there were two or more distinct Z-related pairs of pcmsets, off-sprung from the same pair of parent classes. For example, 4-19(0148) and 5-21(01458) have the following two pairs of Zo-related offspring: 0011448 + 0144458 and 0011488 + 0001458. Each pair contains the same amount of multiplicity, and, since neither parent exhibits T- or I-symmetry, no pcmset is merely a transposition or inversion of its analogue in the other pair.

[15] Pcmultisets will high multiplicity will be shown alternatively as 4-2(0124) <1,4,3,4>, which shows the parent class and the mfunction applied to it, or $0_1 1_4 2_3 4_4$, which shows compactly shows multiplicity of each object.

parent classes, then, is said here to have a **doubling degree** of 1, and the latter, 8. This doubling degree, or d.degree, represents the minimum number of doublings that would have to occur in each of pcsets in order for the resulting pcmsets to be Z-related.

The lowest doubling degree was also sought when discovering the Zo-relation among offspring of parent classes of different cardinality. The d.degree in these cases is represented by two numerals. For example, msetclasses 025 <5,4,3> and 0257 <3,6,2,1> are Z-related with the shared ICV: 19/0, 20, 12, 0, 15, 0. In order for each msetclass to have 12 members, the trichord receives 9 doublings while the tetrachord receives only 8. The d.degree, then is shown as 8/9, the lower of the two values always coming first.

At first blush, the doubling degree might seem to be yet another form of similarity measure. It could indeed be this, but it also serves a more practical purpose. In the course of music analysis, one might apprehend two collections of pcs as traditional pcsets. They may or may not be Z-related. If they are not, typical analysis would at this point disregard the Z-relation as irrelevant to the pcsets at hand. However, a glance at the d.degree of the pair might be helpful, for if it is a low number—say, 2—the doubling in the music might be examined in hopes of finding a Z- (or Zo-, Ze-, Zoe-) relation. If the d.degree is high, one can assume that the two pcsets not only are not Z-related, but their pcmset offspring are quite resistant to Z-relation. In short, the d.degree specifies the degree to which some pairs of pcsets are “almost” Z-related. Some non-Z-related pcsets do, in fact, become Z-related if we view them a pcmsets with just the right doubling.

What follows are the name and brief description of each of the programs used to produce the results in the subsequent sections. The complete BASIC code for each is included in the appendices.[16]

Mfunctionmaker

This program asks the user to input number of objects, the minimum number of elements and the maximum number of elements. The (maximum) number of elements is not permitted to be lower than the number of objects. To represent objects (pcs) 0, 1, and 2, for example, one needs at least three elements: at least one representative of each object. I do not consider possible zero-multiplicity of an object.[17] The program then produces and exports a list of mfunctions given those constraints. Figure 2.11 shows the results after running the program with the following minimal constraints: number of objects = 3; minimum number of elements = 3; and maximum number or elements = 6. The first column represents the number of elements, while the remaining columns comprise the mfunction, giving the multiplicity of each of three objects. This small list of mfunctions, then could be applied to any of the 12 trichord set classes to produce pcmsets containing up to 6 elements. When the program completes the list it also tells the user how many mfunctions were produced.

With this program I produced a list of mfunctions for 3 objects, 4 objects, …, 12 objects, imposing a maximum of 25 elements in each case. The minimum number of elements always was equal to the number of objects; I wanted to include the mfunctions

[16] I wrote these programs in *Chipmunk BASIC*, a free, downloadable version of BASIC available here: http://www.nicholson.com/rhn/basic/. Programmers more experienced than I—and that would be virtually *all* programmers—might find in my code (see appendices) some inefficiencies or redundancies producing more lines of code than necessary and resulting in longer computing times. For that, I apologize, but these programs, such as they are, get the job done.

[17] Zero and even negative multiplicity, however, is an interesting avenue for future research.

Figure 2.11 Sample results produced by *mfunctionmaker.* Constraints: 3 elements, 3 minimum objects, and 6 maximum objects. The first column represents total number of elements. The remaining columns represent multiplicity of the three objects in the parent class; these three columns comprise the mfunctions <x,y,z>.

```
3 , 1 , 1 , 1
4 , 1 , 1 , 2
5 , 1 , 1 , 3
6 , 1 , 1 , 4
4 , 1 , 2 , 1
5 , 1 , 2 , 2
6 , 1 , 2 , 3
5 , 1 , 3 , 1
6 , 1 , 3 , 2
6 , 1 , 4 , 1
4 , 2 , 1 , 1
5 , 2 , 1 , 2
6 , 2 , 1 , 3
5 , 2 , 2 , 1
6 , 2 , 2 , 2
6 , 2 , 3 , 1
5 , 3 , 1 , 1
6 , 3 , 1 , 2
6 , 3 , 2 , 1
6 , 4 , 1 , 1
0 0 0 0 0
```

<1,1,1>; <1,1,1,1>; etc. This produced all the mfunctions I would need to produce all pcmsets (and ICVs) up to 25 elements. Glancing at Figure 2.11, one might notice two loose ends that need to be tied. First, the list of mfunctions are not in a usable order. Before using these lists to produce pcsmsets and ICVs, I sorted them by number of elements (first column.) Second, some of these mfunctions are redundant when applied to inversionally or transpositionally symmetrical parent classes. For example, <2,1,1> and <1,1,2>, when applied to 3-1(012), produce different members of the same msetclass. (Pcmsets 0012 and 0122 are related by T_2I.) This redundancy is not accounted for at this stage; both <1,1,2> and <2,1,1> are necessary when applying to non-symmetrical parent

classes. It is accounted for, however, when ICVs are compared and when Z-relations among offspring of the same parent class are sought.

Vectormaker

I wrote several versions of vectormaker, one for each parent-class cardinality. This program asks the user to input each object (pc). Using the mfunction list produced by mfunctionmaker for that cardinality, it then produces and exports a list of ICVs for all of the off-sprung pcmsets of the requested parent class up to the maximum number or elements requested of mfunctionmaker. (My preliminary calculations used a maximum of 25 elements.) Figure 2.12 provides some sample results. In this case, 5-38 (01258) was the parent class. The first 24 results are shown here, but the program produced 53,130 off-sprung pcmsets and their resulting ICVs. The first column of data represents the total number of elements. The next seven columns represent the seven positions of the ICV. The last five columns represent positions in the mfunction. For example, the third line can be read this way: This pcmset has **6** elements, its ICV is **1**/**214331**, and its mfunction is <**1**,**1**,**1**,**2**,**1**>.

I repeated this process for each of the parent classes—the canonical set classes. This gave me a separate data file, each named for the parent class that produced the pcmsets within. The ICV will be compared in the next step, but the mfunction is included here in order to know exactly which pcmset contains the vector. The mfunction, in this way, stands in for the name of the pcmset. For example, all the pcmsets in Figure 2.12 are offspring of 5-38(01258), but reading the last five columns and applying the mfunction, one can see that the pcmsets, themselves, are as follows: 01258, 012588, 012558, 012258, 011258, 001258, 0125888, etc.

Figure 2.12 Sample results produced by *vectormaker.* Inputted pcs (objects): 0, 1, 2, 5, 8. The first column represents total number of elements. The next seven columns represent the seven positions of the ICV. The last five columns represent positions in the mfunction. (These are the first 24 results out of 53,130 pcmsets with up to 25 elements.)

5 ,0 ,2 ,1 ,2 ,2 ,2 ,1 ,1 ,1 ,1 ,1 ,1
6 ,1 ,2 ,1 ,3 ,3 ,3 ,2 ,1 ,1 ,1 ,1 ,2
6 ,1 ,2 ,1 ,4 ,3 ,3 ,1 ,1 ,1 ,1 ,2 ,1
6 ,1 ,3 ,2 ,3 ,2 ,2 ,2 ,1 ,1 ,2 ,1 ,1
6 ,1 ,4 ,1 ,2 ,3 ,3 ,1 ,1 ,2 ,1 ,1 ,1
6 ,1 ,3 ,2 ,2 ,3 ,3 ,1 ,2 ,1 ,1 ,1 ,1
7 ,3 ,2 ,1 ,4 ,4 ,4 ,3 ,1 ,1 ,1 ,1 ,3
7 ,2 ,2 ,1 ,6 ,4 ,4 ,2 ,1 ,1 ,1 ,2 ,2
7 ,3 ,2 ,1 ,6 ,4 ,4 ,1 ,1 ,1 ,1 ,3 ,1
7 ,2 ,3 ,2 ,4 ,3 ,3 ,4 ,1 ,1 ,2 ,1 ,2
7 ,2 ,3 ,2 ,6 ,3 ,3 ,2 ,1 ,1 ,2 ,2 ,1
7 ,3 ,4 ,3 ,4 ,2 ,2 ,3 ,1 ,1 ,3 ,1 ,1
7 ,2 ,4 ,1 ,3 ,4 ,5 ,2 ,1 ,2 ,1 ,1 ,2
7 ,2 ,4 ,1 ,4 ,5 ,4 ,1 ,1 ,2 ,1 ,2 ,1
7 ,2 ,6 ,2 ,3 ,3 ,3 ,2 ,1 ,2 ,2 ,1 ,1
7 ,3 ,6 ,1 ,2 ,4 ,4 ,1 ,1 ,3 ,1 ,1 ,1
7 ,2 ,3 ,2 ,3 ,5 ,4 ,2 ,2 ,1 ,1 ,1 ,2
7 ,2 ,3 ,2 ,4 ,4 ,5 ,1 ,2 ,1 ,1 ,2 ,1
7 ,2 ,4 ,4 ,3 ,3 ,3 ,2 ,2 ,1 ,2 ,1 ,1
7 ,2 ,6 ,2 ,2 ,4 ,4 ,1 ,2 ,2 ,1 ,1 ,1
7 ,3 ,4 ,3 ,2 ,4 ,4 ,1 ,3 ,1 ,1 ,1 ,1
8 ,6 ,2 ,1 ,5 ,5 ,5 ,4 ,1 ,1 ,1 ,1 ,4
8 ,4 ,2 ,1 ,8 ,5 ,5 ,3 ,1 ,1 ,1 ,2 ,3
8 ,4 ,2 ,1 ,9 ,5 ,5 ,2 ,1 ,1 ,1 ,3 ,2

vectorcomparer

This, the third program in the process, uses the lists of pcmsets (msetclasses when in prime form) and ICVs to search for pairs of msetclasses with matching ICVs. If the msetclasses in the pair come from parent classes of the same cardinality, they are Z-related. If from a different cardinality, Zo-related. There are several versions of this program: one for comparing trichord parent classes, another for comparing tetrachord parent classes, yet another for comparing a trichord parent class with a tetrachord parent

class, etc. The program asks the user to input the two parent classes. It then pulls the requisite data from the lists produced by *vectormaker* and systematically compares the pcmsets, one at a time, moving progressively from low number of elements to high. When it finds matching ICVs, the program stops and displays the shared ICV and the different pcmsets' mfunctions. It then asks if the user would like to continue. As described above, I recorded only the earliest found match—I was seeking those pairs with the lowest doubling degree—but I would continue the program at this point. I had to make sure I'd exhausted all distinct pairs of Z-related pcmsets at that low d.degree. (The program constantly displays the number of elements in the pcmsets it is currently comparing—"6 elements…6 elements…7 elements…etc."—to let the user know when it has moved on to the next higher d.degree.) If one of the parent classes being compared was T- or I-symmetrical, I had to expect at this stage multiple instances of the same pair of pcmsets. For example, the following two pcmsets are Z-related:

0148 <1,2,1,2> (= 011488) ***and*** 0158 <2,1,1,2> (= 001588) **ICV** = 2/202540

The program provided those results and these, as well:

0148 <1,2,1,2> (=011488) ***and*** 0158 <1,2,2,1> (=011558) **ICV** = 2/202540

With little difficulty one can see that two offspring of 0158 are related by T_1I; they belong to the same msetclass. The first of the two, <2,1,1,2> or 001588, is preferred as the prime form.[18] These redundancies were quickly identified, and only the prime forms were recorded. Figure 2.13 shows some sample results.

The program can be used to compare a parent class to itself, but while the redundancies *between* parent classes due to symmetry, described above, are easy to

[18] One can spot this relation quickly by first recognizing that the parent class, 4-20 (0158), is, itself, inversionally symmetrical, then by rotating and retrograding the one of the mfunctions at the correct position. The two mfunctions (and the multiplicities they represent) will be identical.

manage, the redundancies *within* a parent class become overwhelming. Every pcmset gets compared with itself and automatically produces a "match." Therefore, I wrote several modified versions of *vectorcomparer* that take out these redundancies as well as those caused by T- or I-symmetry. (There is a different version for each degree of T- or I-symmetry of the parent class.)

Figure 2.13 Sample results produced by *vectorcomparer.* Inputted hexachord parent 02468T and pentachord parent 02468. Suffix ".txt" is added to correspond to data file. "More" would have been requested in this case until all 9-element mfunctions were compared. (Most of theme represented inversions of one another, i.e., <1,1,1,1,1,4>, <1,1,1,1,4,1>, <1,1,1,4,1,1>, etc. are applied to 02468T producing T_nI-related offspring.) Successful matches among higher cardinalities are ignored here in favor of the lowest d.degree.

[…]
[…*running, reporting number of elements in compared mfunctions*]
[…]
9 elements in 02468T.txt and 9 elements in 02468.txt
9 elements in 02468T.txt and 9 elements in 02468.txt
9 elements in 02468T.txt and 9 elements in 02468.txt
9 elements in 02468T.txt and 9 elements in 02468.txt
9 elements in 02468T.txt and 9 elements in 02468.txt
9 elements in 02468T.txt and 9 elements in 02468.txt

02468T.txt with mfunction <1 ,1 ,1 ,1 ,1 ,4 >
and
02468.txt with mfunction <1 ,3 ,1 ,3 ,1 >
have vector: 6 0 12 0 12 0 6

More? (y/n)

These three programs could be modified to serve other purposes—*vectorcomparer* is modified for finding Ze- and Zoe-relations, for example—but the intended ends already have been met. They have discovered the Z- and Zo-related pairs and have provided the ICV and doubling degree. The following section presents these results.

2.2.4 Z- and Zo-Related Pitch-Class Multisets

After a systematic pairing of all parent classes within each of the different interval profiles, many Z- (and Zo-) related multisetclasses were identified, and the amount of doubling required to produce them (the d.degree) varied widely. What follows is a series of tables, each essentially an elaboration of one of the interval profile groups found in Figure 2.10. Each parent class with a particular interval profile is compared to the others that share the profile. The value given in the grid represents the d.degree at which a Z- (or Zo-) relation is found.[19]

It is reasonable to surmise that some parent classes are unable to produce Z-related offspring no matter how many pc doublings are made. There may be an inherent intervallic construction in one or the other parent that precludes a match; one pc doubling might produce the right number of one interval while simultaneously skewing the number of another interval. It may also be the case that a match is always possible, so long as the doubling degree is high enough. In this respect, finding a Z-related pair might be analogous to the finding of a lowest common denominator but with several denominators! Proposing and proving either scenario is beyond the scope of this dissertation, but the results herein might assist one who endeavors to do so.

The first set of tables is found in Figure 2.14. It includes the interval profiles in profile-group 2. These are the profiles shared by only two or three parent classes each.[20] Among these pair-wise combinations is the unsurprising finding that the pairs with the

[19] Because the calculations are ongoing, other values may appear in the grid. F*x* represents failure to find a Z-relation up to a d.degree of x – cardinality of parent(s). (*x* = number of elements). The higher *x* is, the longer the time required for calculation. Therefore some calculations were suspended and postponed.
[20] Not forgotten, profile group 1 consists of those interval profiles exhibited by only one parent class and, therefore, is considered in section 2.2.7, which discusses Z-related offspring of a single parent.

lowest d.degrees are those where one parent is a subset of the other. Some pairs have failed to produce a match even with as many as 24 elements in the multiset classes.

Figure 2.14 D.degree for Z- and Zo-relations in profile group 2. The returned values are the degrees of doubling required in the parent classes in order to produce Z- or Zo-related multiset classes.

XXXXoX	02346
0236	FAIL 23

XXXooo	0123
013	8/9

XoXXXX	01478
0147	FAIL 16

XoooXX	0167
016	2/3

oXXXXX	02469
0258	FAIL 24

oXXoXo	0257
025	8/9

ooXooX	0369
036	2/3

oooXoo	048
04	3/4

XXXXoo	0124	0134	01234
0124	-	-	-
0134	8	-	-
01234	5/6	9/10	-

oXXXXo	0247	0358	02479
0247	-	-	-
0358	8	-	-
02479	5/6	9/10	-

Figure 2.15 D.degree for Z- and Zo-relations in profile group 3. The returned values are the degrees of doubling required in the parent classes in order to produce Z- or Zo-related multiset classes.

XXoXXX	0126	0157	01267	01268	012678
0126	-	-	-	-	-
0157	1	-	-	-	-
01267	6/5	6/5	-	-	-
01268	3/2	3/2	1	-	-
012678	6/8	6/8	2/3	2/3	-

XoXXXo	0145	0148	0158	0347	01458	014589
0145	-	-	-	-	-	-
0148	2	-	-	-	-	-
0158	2	2	-	-	-	-
0347	2	2	2	-	-	-
01458	4/5	2/3	4/5	4/5	-	-
014589	F 10	3/5	F 10	F 10	3/4	-

oXoXoX	026	0246	0248	0268	02468	02468T
026	-	-	-	-	-	-
0246	9/8	-	-	-	-	-
0248	F 25	1	-	-	-	-
0268	2/3	2	2	-	-	-
02468	6/5	2/3	2/3	4/5	-	-
02468T	6/9	5/7	3/5	F 10	2/3	-

XXXXXo	0125	0135	0237	01235	01245	01348	01358	02347	02357	03458	012345	013458	023457	024579
0125	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0135	14	-	-	-	-	-	-	-	-	-	-	-	-	-
0237	1	14	-	-	-	-	-	-	-	-	-	-	-	-
01235	F 15	F 11	F 11	-	-	-	-	-	-	-	-	-	-	-
01245	F 11	F 11	F 14	7	-	-	-	-	-	-	-	-	-	-
01348	4/5	F 11	F 15	8	3	-	-	-	-	-	-	-	-	-
01358	F 11	7/8	10/11	8	3	6	-	-	-	-	-	-	-	-
02347	2/3	F 11	3/2	10	1	4	1	-	-	-	-	-	-	-
02357	F 11	F 11	F 19	2	7	11	7	10	-	-	-	-	-	-
03458	F 11	F 11	4/5	11	6	0	3	4	8	-	-	-	-	-
012345	F 11	F 10	F 10	4/5	F 10	F 10	4/5	F 10	F 10	F 11	-	-	-	-
013458	F 10	3/5	F 10	F 10	3/4	3/4	3/4	3/4	F 10	3/4	4	-	-	-
023457	F 10	F 10	F 10	3/4	F 10	F 10	F 10	4/5	3/4	F 10	2	2	-	-
024579	F 10	F 10	F 10	F 10	4/5	F 10	F 10	F 10	4/5	F 10	4	4	2	-

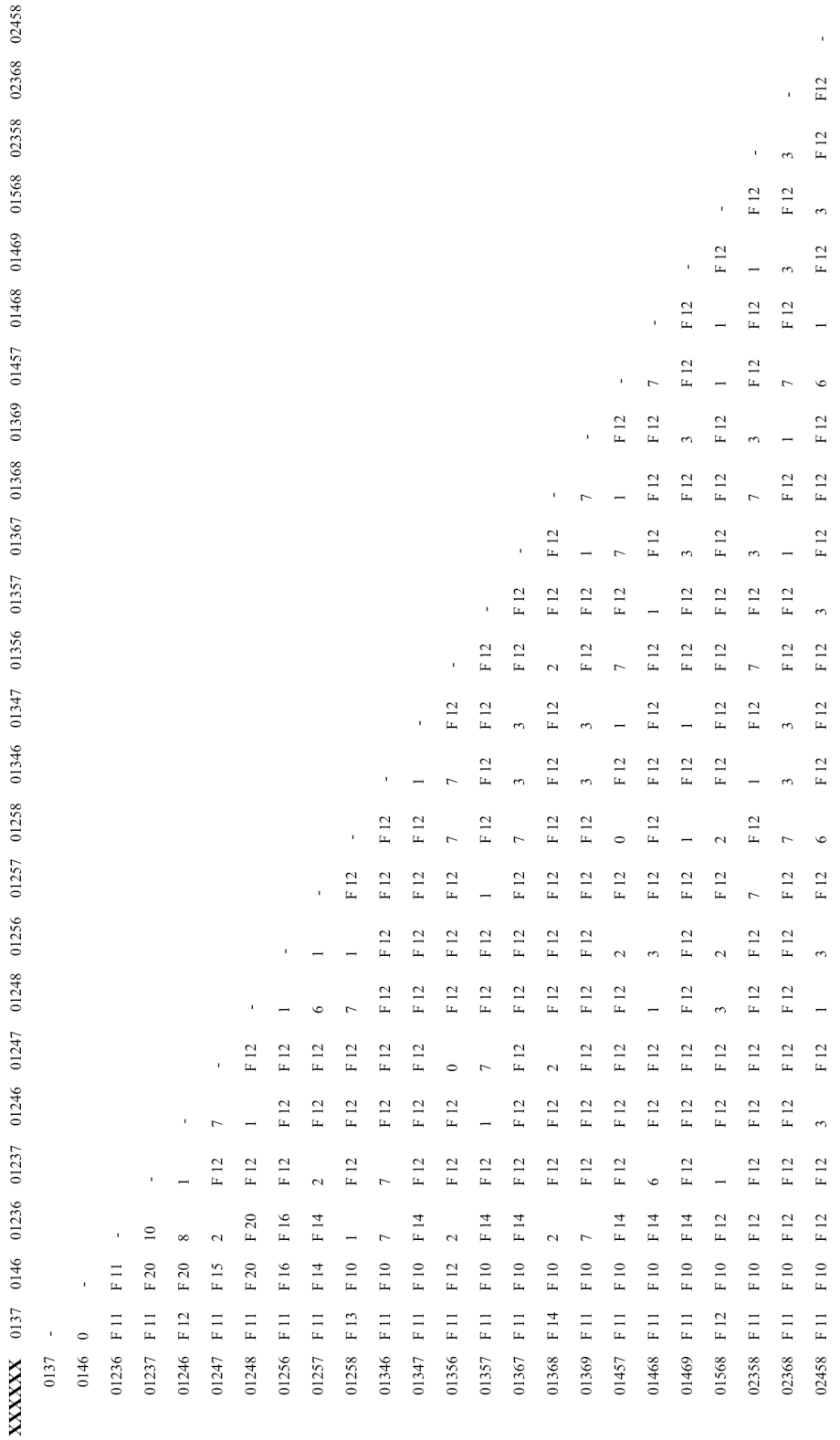
The next set of tables, found in Figure 2.15, includes the interval profiles in profile group 3. This, the second largest profile group, includes the profiles shared by as many as fourteen parent classes. Here we find the Z-related set classes 01348 and 03458. As expected, their d.degree is 0. No doubling is necessary to find a Z-relation; the parent classes, themselves, are Z-related. There are quite a few other pairs with that profile, however, that require minimal doubling to bring forth the Z-relation. D.degrees of 1, 2, and 3 are relatively plentiful. In a music-analytical scenario, doubling degrees of up to 3 are quite plausible, whether through instrumental doubling, repetition, or more straightforward doubling. Higher d.degrees remain a theoretical curiosity with probably rare analytical application.

Figure 2.16 is the table for profile group 4. This group consists of one profile: XXXXXX. In a class by itself, this profile includes 2 tetrachord parents, 22, pentachord parents, 43 hexachord parents, and all parents of higher cardinality. This figure is limited to a comparison of tetrachord and pentachord parents with this profile. Once again, many pairs of parent classes need only 1, 2, or 3 doublings to achieve Z-relation.[21]

The preceding figures provide only the d.degree at which a Z-relation is found. Figure 2.17 lists the d.degree of every successful combination in these tables but further lists the mfunction of the Z- and Zo-related multiset classes as well as the ICV they share. The related pairs are sorted first by doubling degree, then by profile group, and then by profile. The relation, Z or Zo, is given along with the multiset classes, themselves, which are given as parent class with mfunction. The last two columns show d.degree and ICV.

[21] Results of ongoing calculations into higher numbers of elements and objects can be found at https://wfs.gc.cuny.edu/TRobinson/www/dissertation.htm

Figure 2.16 D.degree for Z- and Zo-relations in profile group 4 (# of objects: 4,5 x 4,5). The returned values are the degrees of doubling required in the parent classes in order to produce Z- or Zo-related multiset classes.



XXXXXX	0137	0146	01236	01237	01246	01247	01248	01256	01257	01258	01346	01347	01356	01357	01367	01368	01369	01457	01468	01469	01568	02358	02368	02458
0137	-																							
0146	0	-																						
01236	F 11	F 11	-																					
01237	F 11	F 20	10	-																				
01246	F 12	F 20	8	1	-																			
01247	F 11	F 15	2	F 12	7	-																		
01248	F 11	F 20	F 20	F 12	1	F 12	-																	
01256	F 11	F 16	F 16	F 12	F 12	F 12	1	-																
01257	F 11	F 14	F 14	2	F 12	F 12	6	1	-															
01258	F 13	F 10	1	F 12	F 12	F 12	7	1	F 12	-														
01346	F 11	F 10	7	7	F 12	F 12	F 12	F 12	F 12	F 12	-													
01347	F 11	F 10	F 14	F 12	F 12	F 12	F 12	F 12	F 12	F 12	1	-												
01356	F 11	F 12	2	F 12	F 12	0	F 12	F 12	F 12	7	7	F 12	-											
01357	F 11	F 10	F 14	F 12	1	7	F 12	F 12	1	F 12	F 12	F 12	F 12	-										
01367	F 11	F 10	F 14	F 12	F 12	F 12	F 12	F 12	F 12	7	3	3	F 12	F 12	-									
01368	F 14	F 10	2	F 12	F 12	2	F 12	F 12	F 12	F 12	F 12	F 12	2	F 12	F 12	-								
01369	F 11	F 10	7	F 12	F 12	F 12	F 12	F 12	F 12	F 12	3	3	F 12	F 12	1	7	-							
01457	F 11	F 10	F 14	F 12	F 12	F 12	F 12	2	F 12	0	F 12	1	7	F 12	7	1	F 12	-						
01468	F 11	F 10	F 14	6	F 12	F 12	1	3	F 12	F 12	F 12	F 12	F 12	1	F 12	F 12	F 12	7	-					
01469	F 11	F 10	F 14	F 12	F 12	F 12	F 12	F 12	F 12	1	F 12	1	F 12	F 12	3	F 12	3	F 12	F 12	-				
01568	F 12	F 10	F 12	1	F 12	F 12	3	2	F 12	2	F 12	F 12	F 12	F 12	F 12	F 12	F 12	1	1	F 12	-			
02358	F 11	F 10	F 12	F 12	F 12	F 12	F 12	F 12	7	F 12	1	F 12	7	F 12	3	7	3	F 12	F 12	1	F 12	-		
02368	F 11	F 10	F 12	F 12	F 12	F 12	F 12	F 12	F 12	7	3	3	F 12	F 12	1	F 12	1	7	F 12	3	F 12	3	-	
02458	F 11	F 10	F 12	F 12	3	F 12	1	3	F 12	6	F 12	F 12	F 12	3	F 12	F 12	F 12	6	1	F 12	3	F 12	F12	-

Figure 2.17 Z- and Zo-related multiset classes. Z- and Zo-related pairs are ranked by d.degree within each profile. Multiset classes are listed by parent class and mfunction, and the shared ICV is in the rightmost column.

Gr.	profile	rel.	multiset class A		multiset class B		d.deg	ICV
3	**XXXXXo**	Z	01348	<1,1,1,1,1>	03458	<1,1,1,1,1>	0	0/2, 1, 2, 3, 2, 0
4	**XXXXXX**	Z	01247	<1,1,1,1,1>	01356	<1,1,1,1,1>	0	0/2, 2, 2, 1, 2, 1
4	**XXXXXX**	Z	01258	<1,1,1,1,1>	01457	<1,1,1,1,1>	0	0/2, 1, 2, 2, 2, 1
4	**XXXXXX**	Z	0137	<1,1,1,1>	0146	<1,1,1,1>	0	0/1, 1, 1, 1, 1, 1
3	**oXoXoX**	Z	0246	<2,1,1,1>	0248	<1,2,1,1>	1	1/0, 4, 0, 3, 0, 2
3	**XXoXXX**	Z	0126	<1,1,1,2>	0157	<1,2,1,1>	1	1/2, 1, 0, 2, 2, 2
3	**XXoXXX**	Z	01267	<1,1,2,1,1>	01268	<1,2,1,1,1>	1	1/4, 2, 0, 2, 4, 2
3	**XXXXXo**	Z	01245	<1,1,1,1,2>	02347	<1,1,2,1,1>	1	1/4, 2, 3, 3, 2, 0
3	**XXXXXo**	Z	0125	<1,1,1,2>	0237	<1,1,2,1>	1	1/2, 1, 2, 2, 2, 0
3	**XXXXXo**	Z	01358	<2,1,1,1,1>	02347	<1,1,1,1,2>	1	1/2, 2, 3, 3, 4, 0
4	**XXXXXX**	Z	01236	<1,1,1,1,2>	01258	<1,1,2,1,1>	1	1/3, 2, 3, 2, 2, 2
4	**XXXXXX**	Z	01237	<1,1,1,2,1>	01246	<1,2,1,1,1>	1	1/4, 3, 2, 2, 2, 1
4	**XXXXXX**	Z	01237	<1,1,1,1,2>	01568	<1,1,1,2,1>	1	1/3, 2, 1, 2, 4, 2
4	**XXXXXX**	Z	01246	<2,1,1,1,1>	01248	<1,1,2,1,1>	1	1/3, 4, 1, 3, 1, 2
4	**XXXXXX**	Z	01246	<1,1,1,1,2>	01357	<1,2,1,1,1>	1	1/2, 4, 1, 3, 2, 2
4	**XXXXXX**	Z	01248	<1,2,1,1,1>	01256	<1,1,2,1,1>	1	1/4, 2, 2, 3, 2, 1
4	**XXXXXX**	Z	01248	<1,1,1,1,2>	01468	<2,1,1,1,1>	1	1/2, 2, 1, 5, 2, 2
4	**XXXXXX**	Z	01248	<1,1,1,2,1>	02458	<1,1,2,1,1>	1	1/2, 3, 2, 5, 1, 1
4	**XXXXXX**	Z	01256	<2,1,1,1,1>	01257	<1,2,1,1,1>	1	1/4, 2, 1, 2, 3, 2
4	**XXXXXX**	Z	01256	<1,1,1,2,1>	01258	<1,2,1,1,1>	1	1/2, 1, 2, 3, 3, 1
4	**XXXXXX**	Z	01257	<1,1,1,2,1>	01357	<2,1,1,1,1>	1	1/2, 3, 2, 2, 4, 1
4	**XXXXXX**	Z	01258	<1,1,1,2,1>	01469	<1,2,1,1,1>	1	1/2, 1, 4, 3, 3, 1
4	**XXXXXX**	Z	01346	<2,1,1,1,1>	01347	<1,2,1,1,1>	1	1/3, 2, 4, 2, 1, 2
4	**XXXXXX**	Z	01346	<1,1,1,1,2>	02358	<1,2,1,1,1>	1	1/2, 3, 4, 1, 2, 2
4	**XXXXXX**	Z	01347	<2,1,1,1,1>	01457	<1,1,2,1,1>	1	1/3, 1, 4, 3, 2, 1
4	**XXXXXX**	Z	01347	<1,1,1,1,2>	01469	<2,1,1,1,1>	1	1/2, 1, 4, 3, 2, 2
4	**XXXXXX**	Z	01357	<1,1,1,1,2>	01468	<1,1,1,2,1>	1	1/1, 4, 1, 3, 3, 2
4	**XXXXXX**	Z	01367	<1,1,2,1,1>	01369	<1,2,1,1,1>	1	1/2, 2, 4, 2, 2, 2
4	**XXXXXX**	Z	01367	<1,1,2,1,1>	02368	<1,1,2,1,1>	1	1/2, 2, 4, 2, 2, 2
4	**XXXXXX**	Z	01368	<2,1,1,1,1>	01457	<1,1,1,1,2>	1	1/2, 2, 3, 2, 3, 2
4	**XXXXXX**	Z	01369	<1,2,1,1,1>	02368	<1,1,2,1,1>	1	1/2, 2, 4, 2, 2, 2
4	**XXXXXX**	Z	01457	<2,1,1,1,1>	01568	<1,1,2,1,1>	1	1/3, 1, 2, 3, 4, 1
4	**XXXXXX**	Z	01468	<1,2,1,1,1>	01568	<1,1,1,1,2>	1	1/2, 2, 2, 3, 4, 1
4	**XXXXXX**	Z	01468	<1,1,2,1,1>	02458	<2,1,1,1,1>	1	1/1, 3, 2, 5, 2, 1
4	**XXXXXX**	Z	01469	<1,1,1,2,1>	02358	<1,1,1,1,2>	1	1/1, 2, 4, 2, 3, 2
3	**oXoXoX**	Z	0246	<2,1,1,2>	0268	<2,2,1,1>	2	2/0, 5, 0, 4, 0, 4
3	**oXoXoX**	Z	0248	<1,2,1,2>	0268	<2,1,1,2>	2	2/0, 4, 0, 5, 0, 4
3	**XoXXXo**	Z	0145	<2,1,2,1>	0148	<2,2,1,1>	2	2/4, 0, 2, 5, 2, 0
3	**XoXXXo**	Z	0145	<2,1,1,2>	0158	<2,2,1,1>	2	2/4, 0, 1, 4, 4, 0
3	**XoXXXo**	Z	0145	<1,2,2,1>	0347	<1,2,2,1>	2	2/4, 0, 4, 4, 1, 0
3	**XoXXXo**	Z	0148	<1,2,1,2>	0158	<2,1,1,2>	2	2/2, 0, 2, 5, 4, 0
3	**XoXXXo**	Z	0148	<1,2,2,1>	0347	<1,2,2,1>	2	2/2, 0, 4, 5, 2, 0
3	**XoXXXo**	Z	0158	<1,1,2,2>	0347	<2,1,1,2>	2	2/1, 0, 4, 4, 4, 0

gr.	profile	rel.	multiset class A		multiset class B		d.deg	ICV
3	**XXXXXo**	Z	01235	<1,1,1,1,3>	02357	<1,1,3,1,1>	2	3/3, 5, 4, 3, 3, 0
3	**XXXXXo**	Z	01235	<2,1,1,1,2>	02357	<1,2,2,1,1>	2	2/4, 5, 4, 2, 4, 0
3	**XXXXXo**	Z	012345	<3,1,1,1,1,1>	023457	<1,1,3,1,1,1>	2	3/7, 6, 5, 4, 3, 0
3	**XXXXXo**	Z	013458	<1,1,2,1,2,1>	023457	<2,1,1,2,1,1>	2	2/5, 6, 5, 5, 5, 0
3	**XXXXXo**	Z	013458	<1,2,2,1,1,1>	023457	<2,1,1,2,1,1>	2	2/5, 6, 5, 5, 5, 0
3	**XXXXXo**	Z	023457	<3,1,1,1,1,1>	024579	<1,1,3,1,1,1>	2	3/3, 6, 5, 4, 7, 0
4	**XXXXXX**	Z	01236	<1,1,1,2,2>	01247	<1,2,1,2,1>	2	2/4, 3, 6, 2, 2, 2
4	**XXXXXX**	Z	01236	<1,2,1,1,2>	01356	<2,2,1,1,1>	2	2/5, 3, 3, 2, 4, 2
4	**XXXXXX**	Z	01236	<1,1,1,1,3>	01368	<3,1,1,1,1>	2	3/3, 2, 4, 3, 3, 3
4	**XXXXXX**	Z	01237	<1,1,1,2,2>	01257	<1,2,1,2,1>	2	2/4, 3, 2, 4, 4, 2
4	**XXXXXX**	Z	01247	<1,1,1,2,2>	01368	<2,1,2,1,1>	2	2/2, 3, 6, 2, 4, 2
4	**XXXXXX**	Z	01256	<1,1,2,2,1>	01457	<1,1,2,2,1>	2	2/5, 2, 4, 4, 3, 1
4	**XXXXXX**	Z	01256	<2,1,1,2,1>	01568	<1,1,2,2,1>	2	2/5, 2, 2, 3, 5, 2
4	**XXXXXX**	Z	01258	<2,1,1,2,1>	01568	<1,1,2,1,2>	2	2/3, 2, 4, 4, 5, 1
4	**XXXXXX**	Z	01356	<2,1,1,2,1>	01368	<2,2,1,1,1>	2	2/4, 3, 3, 2, 5, 2
2	**ooXooX**	Zo	036	<2,2,2>	0369	<3,1,1,1>	2/3	3/0, 0, 8, 0, 0, 4
2	**XoooXX**	Zo	016	<2,2,2>	0167	<3,1,1,1>	2/3	3/4, 0, 0, 0, 4, 4
3	**oXoXoX**	Zo	02468	<2,1,2,2,1>	02468T	<3,1,1,1,1,1>	2/3	3/0, 10, 0, 10, 0, 5
3	**oXoXoX**	Zo	0246	<2,2,1,2>	02468	<1,3,1,1,1>	2/3	3/0, 8, 0, 6, 0, 4
3	**oXoXoX**	Zo	0248	<2,2,1,2>	02468	<3,1,1,1,1>	2/3	3/0, 6, 0, 8, 0, 4
3	**oXoXoX**	Zo	0248	<2,2,2,1>	02468	<1,1,3,1,1>	2/3	3/0, 8, 0, 8, 0, 2
3	**oXoXoX**	Zo	026	<2,2,2>	0268	<3,1,1,1>	2/3	3/0, 4, 0, 4, 0, 4
3	**XoXXXo**	Zo	0148	<1,2,2,2>	01458	<1,1,1,1,3>	2/3	3/2, 0, 4, 8, 4, 0
3	**XoXXXo**	Zo	0148	<2,2,2,1>	01458	<1,1,3,1,1>	2/3	3/4, 0, 4, 8, 2, 0
3	**XoXXXo**	Zo	0148	<2,2,1,2>	01458	<3,1,1,1,1>	2/3	3/4, 0, 2, 8, 4, 0
3	**XXoXXX**	Zo	01267	<2,1,2,2,1>	012678	<3,1,1,1,1,1>	2/3	3/6, 4, 0, 4, 6, 5
3	**XXoXXX**	Zo	01267	<1,2,2,1,2>	012678	<1,3,1,1,1,1>	2/3	3/8, 2, 0, 2, 8, 5
3	**XXoXXX**	Zo	01268	<2,2,1,2,1>	012678	<3,1,1,1,1,1>	2/3	3/6, 4, 0, 4, 6, 5
3	**XXoXXX**	Zo	0126	<2,1,2,2>	01268	<3,1,1,1,1>	2/3	3/4, 4, 0, 4, 2, 4
3	**XXoXXX**	Zo	0157	<1,2,2,2>	01268	<1,1,1,3,1>	2/3	3/2, 4, 0, 4, 4, 4
3	**XXXXXo**	Zo	0125	<1,2,2,2>	02347	<1,1,3,1,1>	2/3	3/6, 2, 4, 4, 2, 0
3	**XXXXXo**	Zo	0237	<2,1,2,2>	02347	<1,1,1,1,3>	2/3	3/2, 2, 4, 4, 6, 0
4	**XXXXXX**	Zo	01236	<2,1,1,2,2>	012369	<3,1,1,1,1,1>	2/3	3/5, 4, 8, 2, 2, 4
4	**XXXXXX**	Zo	01237	<1,2,2,1,2>	012378	<1,3,1,1,1,1>	2/3	3/8, 4, 1, 2, 6, 4
3	**XXXXXo**	Z	01245	<1,2,1,1,3>	01348	<3,2,1,1,1>	3	4/7, 2, 5, 7, 3, 0
3	**XXXXXo**	Z	01245	<2,1,1,1,3>	01358	<3,2,1,1,1>	3	4/6, 3, 4, 5, 6, 0
3	**XXXXXo**	Z	01245	<3,1,1,1,2>	01358	<2,3,1,1,1>	3	4/6, 4, 3, 5, 6, 0
3	**XXXXXo**	Z	01358	<3,1,1,1,2>	03458	<3,1,1,2,1>	3	4/3, 2, 5, 7, 7, 0
4	**XXXXXX**	Z	01246	<2,1,1,3,1>	02458	<1,2,3,1,1>	3	4/3, 8, 3, 7, 1, 2
4	**XXXXXX**	Z	01248	<1,2,1,1,3>	01568	<3,1,1,1,2>	3	4/4, 2, 2, 7, 6, 3
4	**XXXXXX**	Z	01256	<1,1,2,1,3>	01468	<3,2,1,1,1>	3	4/6, 2, 2, 7, 4, 3
4	**XXXXXX**	Z	01256	<1,1,3,2,1>	02458	<1,1,2,3,1>	3	4/6, 3, 6, 5, 3, 1
4	**XXXXXX**	Z	01346	<2,2,1,1,2>	01367	<2,2,2,1,1>	3	3/5, 4, 6, 2, 4, 4
4	**XXXXXX**	Z	01346	<2,1,2,1,2>	01369	<2,2,2,1,1>	3	3/4, 4, 9, 2, 2, 4
4	**XXXXXX**	Z	01346	<2,1,1,2,2>	02368	<2,2,2,1,1>	3	3/4, 5, 6, 4, 2, 4
4	**XXXXXX**	Z	01347	<2,2,1,1,2>	01367	<1,1,2,2,2>	3	3/5, 2, 6, 4, 4, 4

gr.	profile	rel.	multiset class A		multiset class B		d.deg	ICV
4	**XXXXXX**	Z	01347	<1,2,1,2,2>	01369	<2,2,1,1,2>	3	3/4, 2, 9, 4, 2, 4
4	**XXXXXX**	Z	01347	<1,2,2,1,2>	02368	<1,2,2,2,1>	3	3/4, 4, 6, 5, 2,4
4	**XXXXXX**	Z	01357	<1,1,3,1,2>	02458	<3,2,1,1,1>	3	4/1, 8, 3, 7, 3, 2
4	**XXXXXX**	Z	01367	<2,1,2,1,2>	01469	<2,2,1,2,1>	3	3/4, 2, 6, 4, 5, 4
4	**XXXXXX**	Z	01367	<1,2,2,2,1>	02358	<1,2,2,1,2>	3	3/4, 4, 6, 2, 5, 4
4	**XXXXXX**	Z	01369	<1,2,1,2,2>	01469	<2,1,1,2,2>	3	3/2, 2, 9, 4, 4, 4
4	**XXXXXX**	Z	01369	<1,2,2,2,1>	02358	<1,2,1,2,2>	3	3/2, 4, 9, 2, 4, 4
4	**XXXXXX**	Z	01469	<2,1,2,2,1>	02368	<2,1,2,1,2>	3	3/2, 4, 6, 5, 4, 4
4	**XXXXXX**	Z	01568	<1,1,2,1,3>	02458	<2,1,1,3,1>	3	4/3, 3, 6, 5, 6, 1
4	**XXXXXX**	Z	02358	<2,2,1,1,2>	02368	<1,1,2,2,2>	3	3/2, 5, 6, 4, 4, 4
2	**oooXoo**	Zo	04	<3,3>	048	<4,1,1>	3/4	6/0, 0, 0, 9, 0, 0
3	**XoXXXo**	Zo	01458	<1,3,1,3,1>	014589	<4,1,1,1,1,1>	3/4	6/6, 0, 6, 12, 6, 0
3	**XoXXXo**	Zo	01458	<2,2,2,2,1>	014589	<3,2,1,1,1,1>	3/4	4/8, 0, 6, 12, 6, 0
3	**XoXXXo**	Zo	01458	<2,2,2,1,2>	014589	<3,1,2,1,1,1>	3/4	4/6, 0, 6, 14, 6, 0
3	**XoXXXo**	Zo	01458	<2,2,1,2,2>	014589	<3,1,1,2,1,1>	3/4	4/6, 0, 6, 12, 8, 0
3	**XoXXXo**	Zo	01458	<2,1,2,2,2>	014589	<3,1,2,1,1,1>	3/4	4/6, 0, 6, 14, 6, 0
3	**XoXXXo**	Zo	01458	<1,2,2,2,2>	014589	<3,1,1,1,1,2>	3/4	4/6, 0, 8, 12, 6, 0
3	**XXXXXo**	Zo	01235	<2,2,2,1,2>	023457	<1,2,3,1,1,1,>	3/4	4/10, 8, 6, 4, 4, 0
3	**XXXXXo**	Zo	01235	<2,2,1,2,2>	023457	<1,2,1,3,1,1>	3/4	4/8, 10, 6, 4, 4, 0
3	**XXXXXo**	Zo	01245	<1,3,1,1,3>	013458	<1,1,1,4,1,1>	3/4	6/9, 2, 6, 10, 3, 0
3	**XXXXXo**	Zo	01245	<2,2,1,2,2>	013458	<1,1,2,3,1,1>	3/4	4/10, 4, 6, 8, 4, 0
3	**XXXXXo**	Zo	01348	<2,2,2,1,2>	013458	<2,1,1,1,3,1>	3/4	4/6, 4, 6, 8, 8, 0
3	**XXXXXo**	Zo	01358	<3,1,1,1,3>	013458	<1,1,1,1,1,4>	3/4	6/3, 2, 6, 10, 9, 0
3	**XXXXXo**	Zo	01358	<2,2,1,2,2>	013458	<1,1,2,1,1,3>	3/4	4/4, 4, 6, 8, 10, 0
3	**XXXXXo**	Zo	02347	<2,1,2,2,2>	013458	<3,1,2,1,1,1>	3/4	4/6, 4, 8, 8, 6, 0
3	**XXXXXo**	Zo	02347	<1,1,3,1,3>	013458	<4,1,1,1,1,1>	3/4	6/6, 2, 6, 10, 6, 0
3	**XXXXXo**	Zo	02357	<1,2,2,2,2>	023457	<3,2,1,1,1,1>	3/4	4/4, 10, 6, 4, 8, 0
3	**XXXXXo**	Zo	02357	<2,2,2,1,2>	023457	<3,1,1,1,2,1>	3/4	4/4, 8, 6, 4, 10, 0
3	**XXXXXo**	Zo	03458	<2,2,2,2,1>	013458	<2,3,1,1,1,1>	3/4	4/8, 4, 6, 8, 6, 0
4	**XXXXXX**	Zo	01236	<1,2,2,2,2>	012347	<3,2,1,1,1,1>	3/4	4/10, 6, 6, 4, 4, 2
4	**XXXXXX**	Zo	01236	<2,2,2,1,2>	012368	<1,1,3,2,1,1>	3/4	4/10, 6, 4, 4, 4, 4
4	**XXXXXX**	Zo	01237	<2,2,2,2,1>	012347	<1,2,3,1,1,1>	3/4	4/12, 8, 4, 2, 4, 2
3	**oXoXoX**	Zo	0248	<2,3,2,2>	02468T	<4,1,1,1,1,1>	3/5	6/0, 12, 0, 12, 0, 6
3	**XoXXXo**	Zo	0148	<2,3,2,2>	014589	<4,1,1,1,1,1>	3/5	6/6, 0, 6, 12, 6, 0
3	**XXXXXo**	Zo	0135	<3,2,2,2>	013458	<1,1,4,1,1,1>	3/5	6/6, 8, 6, 4, 6, 0
4	**XXXXXX**	Zo	0137	<3,2,2,2>	012569	<4,1,1,1,1,1,>	3/5	6/6, 4, 6, 4, 6, 4
4	**XXXXXX**	Zo	0137	<3,2,2,2>	013478	<1,4,1,1,1,1>	3/5	6/6, 4, 6, 4, 6, 4
4	**XXXXXX**	Zo	0146	<2,3,2,2>	012569	<4,1,1,1,1,1>	3/5	6/6, 4, 6, 4, 6, 4
4	**XXXXXX**	Zo	0146	<2,3,2,2>	013478	<1,4,1,1,1,1>	3/5	6/6, 4, 6, 4, 6, 4
3	**XXXXXo**	Z	01348	<3,2,2,1,1>	02347	<1,1,2,3,2>	4	5/8, 4, 8, 7, 4, 0
3	**XXXXXo**	Z	02347	<3,1,2,1,2>	03458	<3,2,1,2,1>	4	5/4, 4, 8, 7, 8, 0
3	**XXXXXo**	Z	012345	<3,1,1,1,2,2>	013458	<1,1,2,2,3,1>	4	5/11, 8, 7, 8, 6, 0
3	**XXXXXo**	Z	012345	<3,1,1,1,1,3>	024579	<1,1,3,3,1,1>	4	6/9, 8, 7, 6, 9, 0
3	**XXXXXo**	Z	013458	<1,3,2,1,1,2>	024579	<1,1,2,3,1,2>	4	5/6, 8, 7, 8, 11, 0
3	**oXoXoX**	Zo	0268	<3,2,1,3>	02468	<4,1,1,2,1>	4/5	7/0, 9, 0, 11, 0, 9
3	**oXoXoX**	Zo	0268	<3,2,2,2>	02468	<3,1,1,3,1>	4/5	6/0, 10, 0, 10, 0, 10

gr.	profile	rel.	multiset class A		multiset class B		d.deg	ICV
3	**oXoXoX**	Zo	0268	<3,3,2,1>	02468	<2,1,1,4,1>	4/5	7/0, 11, 0, 9, 0, 9
3	**XoXXXo**	Zo	0145	<2,3,2,2>	01458	<1,1,3,3,1>	4/5	6/10, 0, 6, 10, 4, 0
3	**XoXXXo**	Zo	0145	<3,2,3,1>	01458	<1,1,4,2,1>	4/5	7/9, 0, 6, 11, 3, 0
3	**XoXXXo**	Zo	0145	<3,2,2,2>	01458	<3,3,1,1,1>	4/5	6/10, 0, 4, 10, 6, 0
3	**XoXXXo**	Zo	0145	<3,1,3,2>	01458	<4,2,1,1,1>	4/5	7/9, 0, 3, 11, 6, 0
3	**XoXXXo**	Zo	0158	<3,1,2,3>	01458	<1,2,1,1,4>	4/5	7/3, 0, 6, 11, 9, 0
3	**XoXXXo**	Zo	0158	<2,2,3,2>	01458	<1,3,1,1,3>	4/5	6/4, 0, 6, 10, 10, 0
3	**XoXXXo**	Zo	0158	<3,2,2,2>	01458	<3,1,1,3,1>	4/5	6/6, 0, 4, 10, 10, 0
3	**XoXXXo**	Zo	0158	<3,2,1,3>	01458	<4,1,1,2,1>	4/5	7/6, 0, 3, 11, 9, 0
3	**XoXXXo**	Zo	0347	<3,1,3,2>	01458	<1,1,1,2,4>	4/5	7/3, 0, 9, 11, 6, 0
3	**XoXXXo**	Zo	0347	<3,2,2,2>	01458	<1,1,1,3,3>	4/5	6/4, 0, 10, 10, 6, 0
3	**XoXXXo**	Zo	0347	<3,2,3,1>	01458	<1,2,4,1,1>	4/5	7/6, 0, 9, 11, 3, 0
3	**XoXXXo**	Zo	0347	<2,3,2,2>	01458	<1,3,3,1,1>	4/5	6/6, 0, 10, 10, 4, 0
3	**XXXXXo**	Zo	01235	<1,3,3,1,2>	012345	<2,4,1,1,1,1>	4/5	7/15, 8, 7, 6, 2, 0
3	**XXXXXo**	Zo	01235	<2,3,1,3,1>	012345	<2,1,4,1,1,1>	4/5	7/12, 14, 7, 3, 2, 0
3	**XXXXXo**	Zo	01245	<3,1,2,1,3>	024579	<1,1,4,2,1,1>	4/5	7/8, 8, 7, 6, 9, 0
3	**XXXXXo**	Zo	01358	<3,3,2,1,1>	012345	<4,1,1,1,1,2>	4/5	7/9, 8, 7, 6, 8, 0
3	**XXXXXo**	Zo	02347	<1,2,1,3,3>	023457	<4,1,2,1,1,1>	4/5	7/5, 8, 10, 6, 9, 0
3	**XXXXXo**	Zo	02347	<3,2,3,1,1>	023457	<2,1,4,1,1,1>	4/5	7/9, 8, 10, 6, 5, 0
3	**XXXXXo**	Zo	02357	<1,2,1,3,3>	024579	<1,4,2,1,1,1>	4/5	7/2, 14, 7, 3, 12, 0
3	**XXXXXo**	Zo	02357	<3,1,2,1,3>	024579	<4,1,1,2,1,1>	4/5	7/2, 8, 7, 6, 15, 0
3	**XXXXXo**	Zo	0125	<1,3,2,3>	01348	<4,2,1,1,1>	4/5	7/9, 2, 6, 9, 3, 0
3	**XXXXXo**	Zo	0237	<2,1,3,3>	03458	<4,1,1,2,1>	4/5	7/3, 2, 6, 9, 9, 0
4	**XXXXXX**	Zo	01236	<2,2,2,2,2>	023469	<1,2,3,2,1,1>	4/5	5/12, 8, 8, 4, 4, 4
4	**XXXXXX**	Zo	01237	<1,2,2,3,2>	012468	<1,3,3,1,1,1>	4/5	6/12, 8, 3, 6, 6, 4
2	**oXXXXo**	Zo	0247	<2,3,2,3>	02479	<4,2,1,2,1>	5/6	8/0, 12, 6, 4, 15, 0
2	**XXXXoo**	Zo	0124	<2,3,3,2>	01234	<4,2,2,1,1>	5/6	8/15, 12, 6, 4, 0, 0
3	**oXoXoX**	Zo	026	<2,4,3>	02468	<5,1,1,1,1>	5/6	10/0, 8, 0, 12, 0, 6
3	**oXoXoX**	Zo	026	<3,4,2>	02468	<1,5,1,1,1>	5/6	10/0, 12, 0, 8, 0, 6
3	**XXoXXX**	Zo	0126	<2,4,2,2>	01267	<1,3,4,1,1>	5/6	9/16, 4, 0, 4, 8, 4
3	**XXoXXX**	Zo	0157	<4,2,2,2>	01267	<1,1,4,1,3>	5/6	9/8, 4, 0, 4, 16, 4
3	**oXoXoX**	Zo	0246	<4,2,3,2>	02468T	<5,2,1,1,1,1>	5/7	11/0, 20, 0, 16, 0, 8
3	**XXXXXo**	Z	01245	<2,1,2,2,4>	03458	<2,4,2,2,1>	6	9/12, 8, 10, 8, 8, 0
3	**XXXXXo**	Z	01348	<2,2,4,1,2>	01358	<4,2,2,2,1>	6	9/8, 8, 10, 8, 12, 0
4	**XXXXXX**	Z	01237	<2,1,2,2,4>	01468	<2,4,1,2,2>	6	9/8, 6, 4, 8, 16, 4
4	**XXXXXX**	Z	01248	<2,4,2,1,2>	01257	<2,4,2,2,1>	6	9/16, 6, 4, 8, 8, 4
4	**XXXXXX**	Z	01258	<1,1,4,4,1>	02458	<1,2,1,5,2>	6	12/5, 4, 20, 5, 5, 4
4	**XXXXXX**	Z	01457	<1,1,4,1,4>	02458	<1,2,1,5,2>	6	12/5, 4, 20, 5, 5, 4
4	**XXoXXX**	Zo	0126	<3,4,2,3>	012678	<5,3,1,1,1,1>	6/8	13/20, 6, 0, 6, 12, 9
4	**XXoXXX**	Zo	0126	<3,5,1,3>	012678	<2,6,1,1,1,1>	6/8	16/20, 3, 0, 3, 15, 9
4	**XXoXXX**	Zo	0157	<5,3,1,3>	012678	<2,1,1,1,6,1>	6/8	16/15, 3, 0, 3, 20, 9
4	**XXoXXX**	Zo	0157	<4,3,2,3>	012678	<5,1,1,1,3,1>	6/8	13/12, 6, 0, 6, 20, 9
3	**oXoXoX**	Zo	026	<3,6,3>	02468T	<7,1,1,1,1,1>	6/9	21/0, 18, 0, 18, 0, 9
3	**XXXXXo**	Z	01235	<1,2,2,3,4>	01245	<2,1,4,3,2>	7	11/12, 20, 11, 8, 4, 0
3	**XXXXXo**	Z	01235	<2,2,3,1,4>	01245	<2,1,3,2,4>	7	11/13, 12, 14, 8, 8, 0
3	**XXXXXo**	Z	01245	<3,1,5,1,2>	02357	<1,2,5,3,1>	7	14/10, 20, 11, 5, 6, 0

gr.	profile	rel.	multiset class A		multiset class B		d.deg	ICV
3	**XXXXXo**	Z	01358	<2,2,4,3,1>	02357	<2,1,4,3,2>	7	11/4, 20, 11, 8, 12, 0
3	**XXXXXo**	Z	01358	<4,2,3,2,1>	02357	<3,2,4,1,2>	7	11/8, 12, 14, 8, 13, 0
4	**XXXXXX**	Z	01236	<2,1,3,2,4>	01346	<4,2,1,3,2>	7	11/11, 8, 12, 12, 4, 8
4	**XXXXXX**	Z	01236	<1,1,3,2,5>	01369	<2,5,1,1,3>	7	14/10, 5, 12, 15, 5, 5
4	**XXXXXX**	Z	01237	<4,2,1,4,1>	01346	<2,5,2,2,1>	7	13/14, 12, 16, 4, 5, 2
4	**XXXXXX**	Z	01246	<1,3,3,3,2>	01247	<3,2,3,3,1>	7	10/12, 18, 9, 9, 6, 2
4	**XXXXXX**	Z	01247	<3,1,3,3,2>	01357	<3,2,3,3,1>	7	10/6, 18, 9, 9, 12, 2
4	**XXXXXX**	Z	01248	<1,3,3,2,3>	01258	<3,2,3,1,3>	7	10/12, 9, 6, 11, 9, 9
4	**XXXXXX**	Z	01257	<1,1,4,4,2>	02358	<2,1,5,2,2>	7	13/5, 12, 16, 4, 14, 2
4	**XXXXXX**	Z	01258	<4,2,3,2,1>	01356	<3,4,2,2,1>	7	11/14, 12, 8, 8, 10, 3
4	**XXXXXX**	Z	01258	<3,1,4,2,2>	01367	<1,3,4,2,2>	7	11/7, 12, 12, 8, 8, 8
4	**XXXXXX**	Z	01258	<3,1,3,2,3>	02368	<3,2,3,1,3>	7	10/6, 9, 12, 11, 9, 9
4	**XXXXXX**	Z	01346	<1,2,2,3,4>	01356	<2,3,4,1,2>	7	11/8, 16, 16, 3, 8, 4
4	**XXXXXX**	Z	01356	<3,2,2,4,1>	01457	<2,1,2,4,3>	7	11/10, 12, 8, 8, 14, 3
4	**XXXXXX**	Z	01356	<2,3,4,1,2>	02358	<3,4,2,2,1>	7	11/8, 16, 16, 3, 8, 4
4	**XXXXXX**	Z	01367	<2,3,4,1,2>	01457	<1,2,2,3,4>	7	11/8, 12, 12, 8, 7, 8
4	**XXXXXX**	Z	01368	<5,1,2,1,3>	01369	<1,5,1,2,3>	7	14/5, 5, 12, 15, 10, 5
4	**XXXXXX**	Z	01368	<4,1,2,2,3>	02358	<3,2,2,1,4>	7	11/4, 8, 12, 11, 8
4	**XXXXXX**	Z	01457	<2,3,1,3,3>	01468	<3,3,2,3,1>	7	10/9, 9, 6, 11, 12, 9
4	**XXXXXX**	Z	01457	<1,3,2,3,3>	02368	<1,3,3,3,2>	7	10/9, 9, 12, 11, 6, 9
3	**XXXXXo**	Zo	0135	<2,1,4,5>	01358	<1,2,6,2,1>	7/8	17/2, 24, 8, 5, 10, 0
4	**XXXXXX**	Zo	01237	<4,1,3,3,2>	012357	<1,1,5,2,2,2>	7/8	13/16, 15, 12, 6, 14, 2
4	**XXXXXX**	Zo	01237	<3,2,1,4,3>	012369	<1,5,1,2,2,2>	7/8	13/12, 11, 12, 12, 12, 6
4	**XXXXXX**	Zo	01237	<3,2,1,4,3>	012369	<2,1,5,1,2,2>	7/8	13/12, 11, 12, 12, 12, 6
2	**oXXXXo**	Z	0247	<1,3,4,4>	0358	<2,5,3,2>	8	15/0, 15, 16, 4, 16, 0
2	**XXXXoo**	Z	0124	<1,4,3,4>	0134	<2,3,5,2>	8	15/16, 15, 15, 4, 0, 0
3	**XXXXXo**	Z	01235	<4,2,1,2,4>	01348	<2,4,4,1,2>	8	14/12, 16, 12, 8, 16, 0
3	**XXXXXo**	Z	01235	<2,1,1,3,5>	01358	<2,3,5,1,1>	8	14/6, 20, 11, 5, 10, 0
3	**XXXXXo**	Z	02357	<1,4,4,2,2>	03458	<2,4,2,4,1>	8	14/16, 16, 12, 8, 12, 0
4	**XXXXXX**	Z	01236	<2,2,5,2,2>	01246	<4,4,2,2,1>	8	14/24, 14, 8, 10, 4, 4
2	**oXXoXo**	Zo	025	<5,4,3>	0257	<3,6,2,1>	8/9	19/0, 20, 12, 0, 15, 0
2	**XXXooo**	Zo	013	<3,5,4>	0123	<6,1,3,2>	8/9	19/15, 20, 12, 0, 0, 0
3	**oXoXoX**	Zo	026	<3,5,4>	0246	<6,1,3,2>	8/9	19/0, 15, 0, 20, 0, 12
4	**XXXXXX**	Zo	01237	<3,2,2,4,4>	012368	<1,3,3,2,5,1>	8/9	17/18, 14, 12, 16, 20, 8
4	**XXXXXX**	Zo	01237	<2,3,1,4,5>	012368	<2,2,3,1,6,1>	8/9	20/13, 14, 8, 20, 15, 15
4	**XXXXXX**	Zo	01237	<3,2,3,4,2>	012479	<2,4,4,2,1,1>	8/9	14/24, 17, 12, 8, 12, 4
2	**oXXXXo**	Zo	0358	<4,3,6,1>	02479	<2,1,2,7,2>	9/10	24/0, 18, 18, 4, 27, 0
2	**XXXXoo**	Zo	0134	<4,6,3,1>	01234	<2,7,1,2,2>	9/10	24/27, 18, 18, 4, 0, 0
3	**XXXXXo**	Z	01235	<4,5,1,3,2>	02347	<3,5,4,2,1>	10	20/28, 25, 14, 10, 8, 0
3	**XXXXXo**	Z	02347	<2,5,1,3,4>	02357	<1,4,2,3,5>	10	20/8, 25, 14, 10, 28, 0
4	**XXXXXX**	Z	01236	<1,3,3,6,2>	01237	<3,2,3,6,1>	10	22/30, 21, 18, 6, 6, 2
3	**XXXXXo**	Zo	0237	<4,2,4,5>	01358	<4,2,1,6,2>	10/11	23/8, 8, 16, 20, 30, 0
3	**XXXXXo**	Z	01235	<3,1,2,3,7>	03458	<3,6,1,5,1>	11	28/11, 30, 23, 7, 21, 0
3	**XXXXXo**	Z	01348	<3,5,6,1,1>	02357	<2,3,7,3,1>	11	28/21, 30, 23, 7, 11, 0
3	**XXXXXo**	Z	0125	<3,4,9,2>	0135	<6,8,3,1>	14	46/48, 27, 18, 8, 60
3	**XXXXXo**	Z	0135	<6,1,3,8>	0237	<9,3,2,4>	14	46/6, 27, 18, 48, 0

Organized this way, one quickly can find the pairs exhibiting a low d.degree, which are perhaps the most analytically relevant.

One may notice some pairs of parent classes listed more than once. In these cases, there is more than one pair of Z- or Zo-related multiset classes at the minimum d.degree. These pairs are not T_nI-related to one another; they are independent results. Having the same low d.degree, yet remaining distinct multiset classes, all such pairs are listed here. Not listed, however, are those pairs exhibiting a higher d.degree. In some cases of low d.degrees, there exist pairs as close as the next higher d.degree. This ignored d.degree might even be significantly lower than the d.degree listed other pairs, but listing only the lowest d.degree for each pair is consistent with the purpose of the present task. This is, in fact, an examination of the properties of the parent classes, not a comprehensive listing of every Z or Zo-related multiset class. Not only would such a listing be infinite, one can easily find the Z- or Zo-relation by comparing ICVs. These results, however, may be a starting point for analysts seeking the Z-relation.

Even a casual scan of the preceding data will give one a good sense of many Z-relations lurking below the surface of the canonical set classes. Extended perusal of the data might suggest avenues for further study.

2.2.5 Soderberg's Dual Inversion

The Z-relation has been observed and discussed for many years, but its *raison d'être* is shrouded in a mystery that is reluctant to dissipate. One effort to answer lingering questions about this strange property casts some Z-related sets as the result of

"dual inversion."[22] Stephen Soderberg defines his operation: "*under circumstances to be described* (4.1–3), if some of the elements of A invert with respect to index x while the other elements of A invert with respect to index y, the resulting set B may be neither a simple transposition nor a simple inversion of A, but both sets may nonetheless share the same interval-class vector; that is, A and B may be Z-related." The emphasis is mine; his "circumstances" must always be kept in mind while performing this operation. Fortunately for us, Soderberg created the "Q-grid," which, with a simple set of rules for its use, makes dual inversion a simple process by building these complex "circumstances" into its structure. This section shows how Q-grids enable the easy finding not only of Z-related pc sets and set classes, but also of Z- (and Zo-) related pcmultisets and multiset classes, with no modification whatsoever to the procedure. While I show the strengths of the technique as a means of discovering Z-related pairs, I also discuss the acknowledged limitations of the Q-grid as a means of enumerating them.

Dual inversion, as it described above, might appear simple, but a naïve attempt to apply it without considering Soderberg's restrictions likely will fail. Simply to invert some elements of a pcset by one index and the remaining elements by another is an interesting proposition, and it can lead to collections of transformationally related pcsets, but this is not *dual* inversion. It is perhaps more closely related to *split* transformation.[23]

---

[22] Stephen Soderberg, "Z-Related Sets as Dual Inversions," *Journal of Music Theory* 39/1 (Spring, 1995): 77-100.

[23] Shaugn O'Donnell proposes the split transformation in "Klumpenhouwer Networks, Isography, and the Molecular Metaphor," *Intégral* 12 (1998): 53–80. He also comments on some published conflation of the terms "dual" and "split" in "Embracing Relational Abundance," *Music Theory Online* 13/3 (September 2007). It seems to me that the split-type transformations may be most valuable when the *operation* is of primary analytical interest (as in a network of several transformationally related sets) and the dual-type transformations are valuable when the intervallic *properties* of the sets are of interest. The much discussed "promiscuity" of the former is not problematic when the analyst is chiefly concerned about the transformation itself, while the "chasteness" imposed by Soderberg's restrictions in the latter strengthens not the transformations but the relationships among the comparatively fewer pairs that emerge.

Dual inversion, in order to ensure equivalence in the resulting IC vectors, works as described above, but the specific subsets chosen are subject to restrictions. I will not explain here these restrictions, which Soderberg does formally in his article.[24] Rather, I will jump straight to his Q-grids, which not only embody and make plain those restrictions anyway, but also allow a straightforward introduction to the procedure.

Soderberg defines his Q-grid with the following notation:

$$[m,\phi;x,y] = \bigcup_{\alpha} \left[\begin{array}{c|c} \Phi & I_y\Phi \\ I_x\Phi & I_xI_y\Phi \end{array}\right]$$

Variables are defined as follows:

m = modulo; this represents the number of tones into which an octave is equally divided ($\mathbb{Z}_m$) and will, for our purposes, always be 12. ($\mathbb{Z}_{12}$ represents our common system of twelve pitch classes, 0-11.)

ϕ = divisor; when m = 12, ϕ must, as a divisor of the octave, equal 12, 6, 4, 3, 2, or 1.

X = index x; this is the first index of inversion

y = index y; this is the second index of inversion

Φ = m/ϕ-sided polygon; this simply is a set generated by the interval cycle ϕ. For example if m = 12 and ϕ = 4, Φ = {0, 4, 8}. The grid is reiterated for every transposition of Φ until aggregate is complete. (α = number of such repetitions.)

*Restriction: $x - y = k\phi/2$ (k = 0, +-1, +-2,…)

[24] Soderberg, "Z-Related Sets," p. 92.

A restriction, listed last, is the first of two definitive restrictions in dual inversion. While this one—a restriction on the axes of inversion—affects the Q-grid's constitution, the other—a restriction on subset selection—affects, as we will see below, the way the Q-grid is used. These two restrictions, in conjunction, ensure identical resulting IC vectors.

Figure 2.18 Q-grid [12, 4; 0, 4]

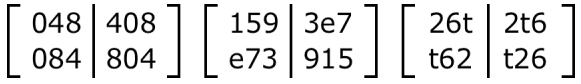


Figure 2.18 contains the Q-grid 12, 4; 0, 4. Given m = 12 and $\phi = 4$, the set Φ, found in the upper left quadrant of each of the three cells, is {0, 4, 8} or one of its transpositions. The contents of these quadrants are inverted both by I_x (in this case, I_0) to yield the pcs in the lower-left quadrants and by I_y (in this case, I_4) to yield the pcs in the upper-right quadrants. The lower-right quadrants, then, are I_0I_4-related to the original Φ. This means that the I_y-relation is found left-to-right, across the centerline of the grid. The I_x-relation is found vertically, each pc above or below its I_y-correspondent. In this way the dual inversion can be seen emerging, one inversion for each of the two dimensions in the grid. To demonstrate dual inversion, then, one selects some pcs to invert in one direction and some to invert in the other. The user makes a "pass" at each cell, selecting some (or no) pcs from each. The selected pcs together comprise the pcset (Q) being dually inverted onto its Q-inversion. ($Q^*=QINV(Q)$[25]). The passes are recorded as P_n where n = cell number 0, 1, 2… and P^*_n = the inverted result. The second important restriction is now imposed, however, on the selecting of these pcs. It is twofold. If a pass

[25] Soderberg, "Z-related Sets," 92.

at a given cell is to use the y (left-right) inversion, one must select *all* of the pcs on the left side or on the right side of the cell. If a pass is to use the x (up-down) inversion, one must select an *equal number* of pcs from the right and left halves.[26] For example, in Q-grid [12, 4; 0, 4] (Figure 2.18), one might, as a pass on the first cell, select {0, 4, 8} (duplications omitted) from the left side. This selection (P_0) would yield the inversion {0, 4, 8}, found on the right side of the grid (P^*_0). A pass on the second cell might take {5, 9}. (This takes, as required, one pc from each side of the grid: pc 5 comes from the upper left, and pc 9, from the lower right.) This selection (P_1) would yield the y-inversion {7, 3}. The final pass, on cell 3, might simply take no pcs at all. (P_2 = {Ø}.) To summarize—in Soderberg's fashion—our selections:

P_0 = {0, 4, 8} P^*_0 = {0, 4, 8}

P_1 = {5, 9} P^*_1 = {7, 3}

P_2 = {Ø} P^*_2 = {Ø}

Q = {4, 5, 8, 9, 0} Q* = {0, 3, 4, 7, 8}

5-21(01458) 5-21(01458)

Clearly, dual inversion has preserved the IC vector of the pcset undergoing the operation, but Q and Q* in this case are simply inversions of one another. The ICV is preserved, but the two sets are not Z-related; they belong to the same set class. This was only one of many possible combinations of passes at this Q-grid, though, and this Q-grid is only one of several possible grids where m = 12.

In his explication of dual inversion, Soderberg did not burden the reader with projections of all available Q-grids. For the following discussion of multiset classes,

[26] It should be apparent to the reader that one must select at least one of each of the two types of passes in order to engage both axes of inversion. Otherwise, simple inversion will result.

Figure 2.19 Q-grids available in $\mathbb{Z}_{12}$

12, 12; 0, 6

0	6
0	6

1	5
e	7

2	4
t	8

3	3
9	9

12, 12; 1, 7

0	7
1	6

2	5
e	8

3	4
t	9

12, 6; 0, 3

06	39
06	93

17	28
e5	t4

12, 6; 0, 6

06	60
06	60

17	5e
e5	71

28	4t
t4	82

39	39
93	93

12, 6; 1, 4

06	4t
17	93

28	28
e5	e5

12, 6; 1, 7

06	71
17	60

28	5e
e5	82

39	4t
t4	93

12, 4; 0, 2

048	2t6
084	t26

159	195
e73	e37

12, 4; 0, 4

048	408
084	804

159	3e7
e73	915

26t	2t6
t62	t26

12, 4; 1, 3

048	3e7
195	t26

12, 4; 1, 5

048	519
195	804

26t	3e7
e73	t26

12, 3; 0, 3

0369	3096
0963	9036

147t	2e85
e852	t147

12, 3; 1, 4

0369	41t7
1t74	9036

258e	2e85
e852	e258

12, 2; 0, 1

02468t	1e9753
0t8642	e13579

12, 2; 0, 2

02468t	20t864
0t8642	t02468

13579e	1e9753
e97531	e13579

12, 1; 0, 1

0123456789te	10et98765432
0et987654321	et9876543210

however, one projection might be helpful, so Figure 2.19 displays all Q-grids available when m = 12, i.e., in $\mathbb{Z}_{12}$. Each grid is labeled according to [m, ϕ; x, y]. Therefore, the "m" entry in each is 12. They are sorted then according to ϕ-values, which are the six divisors of 12. The "x" entry is either 0 or 1, representing even and odd inversion respectively. Any higher x values simply will be transpositions of one of these. The "y" values vary, given the restriction that the difference between x and y must be a multiple of half of the divisor, i.e., $x - y = k\phi/2$ ($k = 0, +-1, +-2,\ldots$). Some possible y values are omitted here, however, for their resulting Q-grids are simple transpositions of others. What is more, y = 0 is not considered. As Soderberg notes, if x and y both are 0, we're left with simple, not dual, transposition.[27]

The Q-grid is a convenient and productive means of identifying many pairs of Z-related pcsets and set classes. One simply has to make enough passes and combinations of passes to find them among the many that are merely T_n- or T_nI-related.

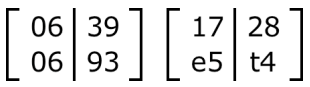
It is perhaps unsurprising that Q-grids, like so many other music-theoretical tools, can accommodate not just pcsets but pc*multi*sets with ease. The "rules" are followed as instructed by Soderberg, but one can repeat a pass, or make a different one, at the same cell in a Q-grid, thus possibly repeating a pc. The following examples demonstrate the kinds of mset pairs one might find after such attempts: Z-related, T_nI-related, Z-related with Z-related parents, and Zo-related.

Working from Q-grid 12, 6; 0, 3, Figure 2.20(a1) shows how multiple passes (P_{1a} and P_{1b}) on the second cell in the grid can duplicate some pcs, creating the pcmultiset Q. P_0 here is a "left-right" pass on the first cell. The two passes on the second cell, P_{1a} and P_{1b}, are "up-down" passes. The resulting Q* in the Q-pair is, like Q, a 5-object

[27] Soderberg, "Z-Related Sets," 95.

Figure 2.20 Q-grid 12, 6; 0, 3 producing five types of result: (a1) a Z-related pair of 5-object hexachords; (a2) another pair using doubling in a single pass rather than two separate passes; (b) a pair of pcmultisets belonging to the same multiset class; (c) Z-related offspring of Z-related parents; (d) a pair of distinct multiset classes from the same parent; and (e) a pair of Zo-related pcmultisets.

(a1)



P_0 = {0, 6} P^*_0 = {3, 9}

P_{1a} = {4, 5} P^*_{1a} = {7, 8}

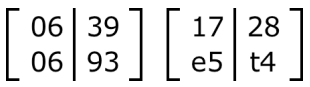
P_{1b} = {4, 7} P^*_{1b} = {5, 8}

Q = {0, 4, 4, 5, 6, 7} Q* = {0, 1, 1, 2, 4, 6}

msc = **012337** msc = **011246**
5-5(01237) <1,1,1,2,1> *5-9(01246) <1,2,1,1,1>*

ICV = 1/4, 3, 2, 2, 2, 1

(a2)



P_0 = {0, 6} P^*_0 = {3, 9}

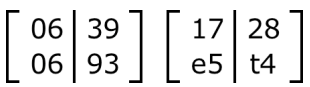
P_1 = {1, 2, 2, 7} P^*_1 = {5, t, t, e}

Q = {0, 1, 2, 2, 6, 7} Q* = {9, t, t, e, 3, 5}

msc = **012267** msc = **011268**
5-7(01267) <1,1,2,1,1> *5-15(01246) <1,2,1,1,1>*

ICV = 1/4, 2, 0, 2, 4, 2

(b)



P_0 = {Ø} P^*_0 = {Ø}

P_{1a} = {1, 2, 7, 8} P^*_{1a} = {4, 5, t, e}

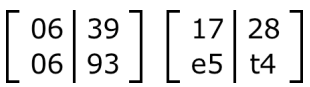
P_{1b} = {1, 5, 7, e} P^*_{1b} = {2, 4, 8, t}

Q = {1,1,2,5,7,8,e} Q* = {2,4,4,5,8,t,t,e}

msc = **01136779** msc = **01136779**
6-30(013679) <1,2,1,1,2,1>

ICV = 2/4, 4, 4, 4, 4, 6

(c)



P_0 = {3, 9} P^*_0 = {0, 6}

P_{1a} = {2, 5, 7, t} P^*_{1a} = {2, 5, 7, t}

P_{1b} = {7, t} P^*_{1b} = {2, 5}

Q = {2,3,5,7,7,9,t,t} Q* = {5,5,6,7,t,0,2,2}

msc = **01355788** msc = **01225579**
6-Z26(013578) <1,1,1,2,1,2> *6-Z48(012579) <1,1,2,2,1,1>*

ICV = 2/3, 5, 5, 5, 7, 1

Figure 2.20 (continued)

(d)

$\left[\begin{array}{c|c} 06 & 39 \\ 06 & 93 \end{array}\right] \left[\begin{array}{c|c} 17 & 28 \\ e5 & t4 \end{array}\right]$

$P_0 = \{\emptyset\}$ $P^*_0 = \{\emptyset\}$

$P_{1a} = \{2, 4, 8, t\}$ $P^*_{1a} = \{5, 7, e, 1\}$

$P_{1b} = \{5, 8\}$ $P^*_{1b} = \{4, 7\}$

$Q = \{2, 4, 5, 8, 8, t\}$ $Q^* = \{e, 1, 4, 5, 7, 7\}$

msc = **023668** msc = **002368**

5-28(02368) <1,1,1,2,1> <1,2,2,2,1> *5-28(02368) <2,1,1,1,1>*

ICV = 1/1, 3, 3, 3, 1, 3

(e)

$\left[\begin{array}{c|c} 06 & 39 \\ 06 & 93 \end{array}\right] \left[\begin{array}{c|c} 17 & 28 \\ e5 & t4 \end{array}\right]$

$P_0 = \{\emptyset\}$ $P^*_0 = \{\emptyset\}$

$P_{1a} = \{5, 7, e, 1\}$ $P^*_{1a} = \{2, 4, 8, t\}$

$P_{1b} = \{t, 1, 1, 2\}$ $P^*_{1b} = \{t, e, e, 2\}$

$Q = \{t,e,1,1,1,2,5,7\}$ $Q^* = \{8,t,t,e,e,2,2,4\}$

msc = **01113479** msc = **02233668**

6-Z49(013479) <1,3,1,1,1,1> *5-28(02368)*

ICV = 3/4, 4, 6, 5, 2, 4

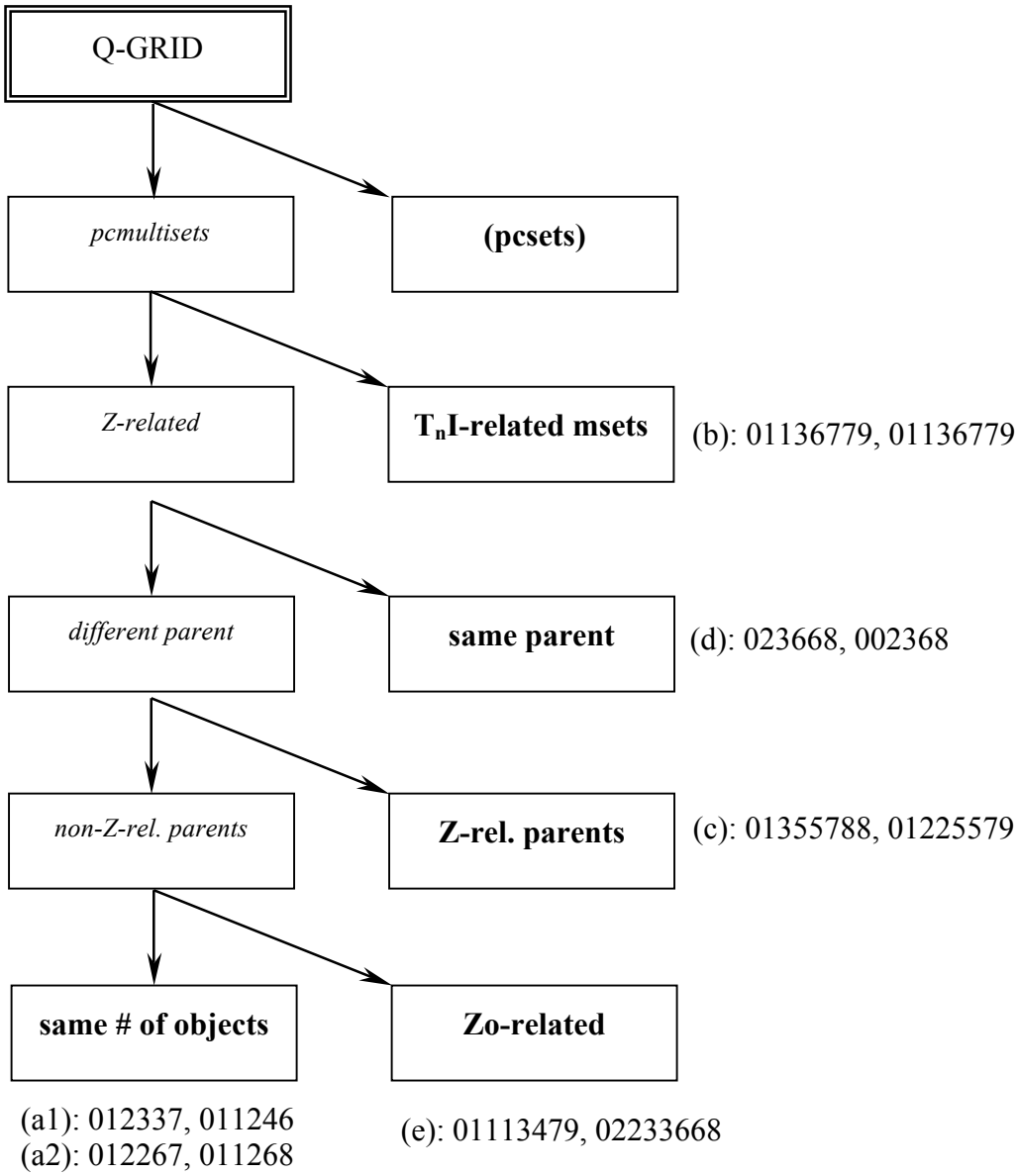
hexachord, and shares the same ICV. The multiset classes to which they belong, however, come from different, non-Z-related, parent classes: 5-5(01237) and 5-9(01246). Figure 2.20(a2) shows another Q-pair, generated just as in the preceding figure but with a slight notational difference in the tabulation. To save time and space, two passes on a single cell are consolidated into one pass. P_1 need not be split into P_{1a} and P_{1b} so long as P_1, as an "up-down" pass takes an equal number of pc representatives—doublings count—from each half. The result found in Figure 2.20(b) is quite common. Here, after three passes, the resulting pcmultisets are T_nI-related, belonging to the same multiset

class. The Q-grids produce many such pairs. As we will see below, most results are ordinary T_nI-related pairs through which one must sift in order to find the real jewels. The passes in Figure 2.20(c) yield a Z-related pair of multiset classes, 01355788 and 01225579. The passes in Figure 2.20 (d) turn up another type of result: a distinct but Z-related multiset classes coming from the same parent. Multiset classes 023668 and 002368 are neither transpositions nor inversions of one another; they are distinct multiset classes. (This type is further discussed in section 2.2.7 below.) Figure 2.20(e) shows a most rare gem, a Zo-related pair: a 5-object octachord and a 6-object octachord. Inspection of the passes chosen will reveal how this came to be. Two passes are made on the second cell. P_{1a} is a "left-right" pass, while P_{1b} is an "up-down" pass. Simply making both types of passes (y-inversion and x-inversion) on the same cell is not sufficient to generate the Zo-relation—figure 2.20(b) makes two such passes. Nor is making an "up-down" pass with a doubled pc on one side and two different pcs on the other—Figures 2.20(a1) and (a2) do this without Zo-related results. It seems the difference in the number of objects in Q and Q* is due to a combination of both types of passes: a "left-right" pass combined with an "up-down" pass that has more doublings on the *same* side that the "left-right" pass selected. In this case P_{1a} {5, 7, e, 1} is combined with P_{1b} {t, 1, 1, 2}, together containing three pc1s. Two get mapped via I_x to e, and one gets mapped via I_y to 3. The reader can make some passes to demonstrate this to satisfaction.

Figure 2.21 sums up. The Q-grid generates a pair of pcmultisets that have the same ICV. The pair may be either T_nI-related or Z-related. If Z-related, the mset classes in the pair may have the same or different parent classes. If the parent classes are

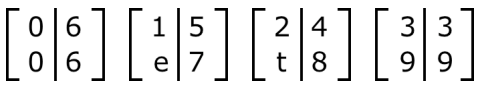
different, they may be Z-related or not. If not, the number of objects may either be the same (if Z-related) or different (if Zo-related). In this figure, each result-type is accompanied by a letter, which corresponds to the appropriate subfigure in Figure 2.20.

Figure 2.21 Types of result produced by a Q-grid. The pairs of mset classes accompanying each type correspond to the examples shown in Figure 2.20 and are labeled accordingly, a-f.



In an attempt to see just how productive the Q-grids can be, I chose one and systematically made all possible combinations of three passes on it. I subjected Q-grid 12, 12; 0, 6 to this scrutiny by first listing and numbering all unique passes (see Figure 2.22) and then combining all possible sets of three passes, while insuring that there was dual inversion, i.e., that there was at least one of each type of inversion, x and y. Of the

Figure 2.22 All possible passes on Q-grid 12, 12; 0, 6



	#	P	P*
y-inversion	1	0	6
	2	6	0
	3	1e	57
	4	57	1e
	5	2t	48
	6	48	2t
	7	39	39
x-inversion	8	06	06
	9	15	e7
	10	17	5e
	11	e5	17
	12	e7	15
	13	157e	157e
	14	24	8t
	15	28	4t
	16	4t	28
	17	8t	24
	18	248t	248t
	19	3	9
	20	39	39
	21	9	3

N.B.: Passes were limited only to those without duplications, as if using the Q-grid in the standard fashion for non-multisets. For this test, duplications are to arise in only the combining of these passes. This explains pass #19, which shows only a single pc3. This represents a selection of pc3 from both the right and the left. When working with multisets, each pc *representative* would appear: 3, 3. The original passes here are treated as pitch *classes*, however.

1,127 resulting pairs, only 707 contain pc duplication. The rest could be treated as canonical set classes. Of those 707, only 96 are instances of Z-related multiset-class pairs. The others are T_nI-related. Furthermore, these 96 are composed only 6 unique pairs, each appearing 16 times.[28] The pairs of multiset classes ultimately found are as follows:

Z-related:	01226	01157	
	00246	02248	
	013367	011369	
	013466	022358	
	013477	001469	
Zo-related:	002266	000268	*Zo-related

In an effort to minimize the number of non-multiset results, I returned to the same Q-grid but with random selections of four passes. Starting with the 21 possible single passes at the Q-grid, I listed the 98 possible combinations of the 7 y-inversion passes and the 14 x-inversion passes. (See Figure 2.22 again.) For each of these 98 pairs of passes I added two more passes at random, making sure that no set of four passes was repeated. This test increased the ratio of multisets to non-multisets returned—75 multiset pairs and 23 non-multiset pairs. Of the 75 multiset pairs, 55 are simply T_nI-related. Of the 20 "successful" finds, 4 pairs are Zo-related and 2 pairs are distinct multiset classes off sprung from the same parent class.

Here arises the first weakness of the Q-grid as enumerator. In the first test case above, 1127 trials produced only 6 Z- (or Zo-) related multiset classes. In the second case, 98 trials discovered 20. A better ratio to be sure, but for systemic, comprehensive enumeration one still must weed out the many T_nI-related pairs. Other similar but more efficient methods might intentionally take same pass twice, three times. Possibly, one

[28] This number, 16, certainly is a result of the inherent symmetry in the square cells of the Q-grid.

could combine those passes that would make at least one doubling. In any case, such methods still would produce many results that are merely T_nI-related. Given the number of Q-grids and possible combinations of passes, the calculations would be of the size and scope of the brute force method of enumeration found in sections 2.2.3 and 2.2.4. Furthermore, while all pairs of pcsets and pcmultisets produced by Q-grids do have the same ICV, some of which are Z-related, Q-grids cannot produce *all* Z-related pairs.[29] It is fair (and important) to assume, of course, that Soderberg never intended for Q-grids to be used in this manner. His thesis is simply that dual inversion *may* produce and, in a sense, explain some Z-related pcsets. Dual inversion, then, may not be an efficient means of mass production, but it is a fascinating operation nonetheless, and by setting apart those msets that can be related by it from those that cannot, one can distinguish some pairs of Z-related pcmultisets from others. The Q-grids, which display dual inversion, are most helpful in the examination of a particular pcset or pcmultiset. With little trouble one can find a Q-grid that will accommodate a given pcset or multiset, X: first look for the Φ-set ({0,6}; {0,4,8}; {0,3,6,9} etc.) that is a subset of X; then check to see if passes with x- or y-inversion are possible; finally, recheck with all T_n/T_nI forms of X.

2.2.6 Ze- and Zoe-Related Pitch-Class Multisets

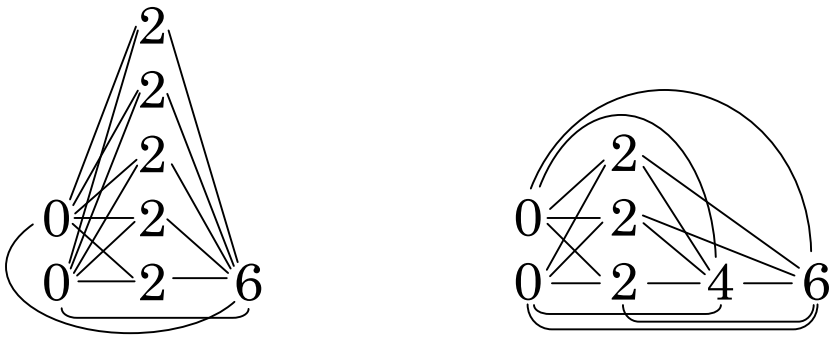
Because any two Z-related sets, whether pc- or pcmulti-, have identical interval-class vectors, they also contain the same number of *elements*. I have shown above that the number of *objects* in the pair, however, may vary. Such a Z-relation was designated

[29] Soderberg points out four pairs of Z-related hexachords that cannot be produced by dual inversion. They "do not 'fit' in any of the possible Q-grids in $\mathbb{Z}_{12}$." He goes on to find commonalities among them which "[suggest] the possibility that there may be undiscovered structures similar to those described by the conditions [for dual inversion]." Perhaps there is a triple inversion or a compound dual inversion just waiting to be uncovered.

"Zo." It seems there is no way for two pcsets or pcmultisets with a different number of elements ever to be Z-related, unless, that is, we compare only the traditional six interval-class positions in the vector. If we ignore the entry in the 0 position of the vector—the unison—we treat pc doubling in a special way: we count the intervals that a doubled pc makes with other pcs, but we discount the unisons that the pc forms with itself. In a sense, we disregard the *pc* doubling but not the *inter-pc* doubling. The pair properly cannot be called Z-related, so such pairs are designated here as **Ze-related**, "e" standing for elements, the aspect in which their ICVs differ. If a Ze-related pair also differs in the number of objects, it is designated here as **Zoe-related**.

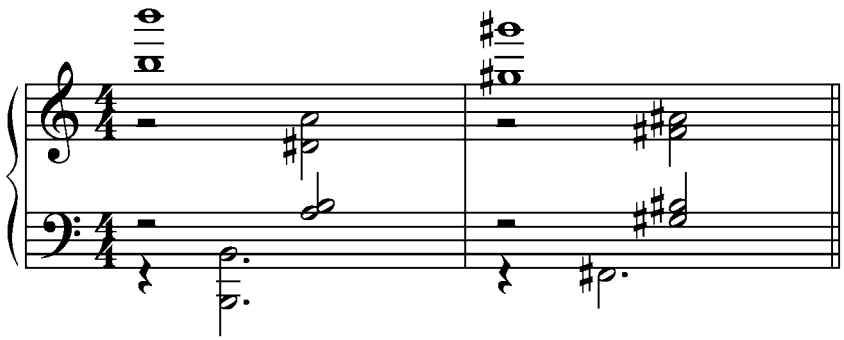
The relative rarity of Ze- and Zoe-related pairs begs further investigation. Figure 2.23 is a representation of multiset classes 025 <2,5,1> and 0246 <2,3,1,1> (or 00222226 and 0022246). Zoe-related, they contain different numbers of objects: eight and seven, respectively. Their ICVs, shown below them, display their similarity; the ignored unison entry is grayed out. The lines, which display the relevant (counted) intervals, emphasize

Figure 2.23 Zoe-related mset classes (displayed). The mset classes are displayed with all constituent intervals shown except the unisons (ic0).



How the Zoe- (or Ze-) relation affects our perspective of the msetclass. In mset class 00222226 we see the collective intensity of the ten ic2s between 0 and 2, but we do not see an equal number of unisons among its pc2s. If their full ICVs were compared, the eleven unisons of one mset class would overpower the other. In contrast, the Ze- (in this case Zoe-) relation allows their particular similarity to come through. Figure 2.24 shows on the staff how these two mset classes might appear in a musical scenario. The displacement of the pc doubling by octaves might support our attempt to overlook the ic0s because, for example, the two ptokens of B in the right hand each form a different pitch interval with each of the remaining pitches in the collection (measure). Their separation in p-space thus highlights the multiple intervals formed with other pitches.

Figure 2.24 Zoe-related mset classes (composed)



{99eeeee3}
026 <2,5,1>
ICV *11* /0, 10, 0, 5, 0, 2

{66888t0}
0246 <2,3,1,1>
ICV *4* /0, 10, 0, 5, 0, 2

The computer programs written for enumeration of the Z/Zo-related pairs, with only minor modifications, are equally suited for seeking Ze/Zoe-related ones. Figures 2.25-2.27 display the findings of these modified programs. The interval profile returns as

the primary sorting criterion, as even Ze/Zoe-related msetclasses must contain the same non-unison intervals. Because the number of unisons is ignored, a pcmset with only one representative of each of its objects (indistinguishable on the surface from a canonical set class[30]) can be Ze/Zoe-related to a pcmset with at least one multiplicity >1. For example,

Figure 2.25 D.degree for Ze- and Zoe-relations in profile group 2. The returned values are the degrees of doubling required in the parent classes in order to produce Ze- or Zoe-related multiset classes.

XXXXoX	02346
0236	F 11

XXXooo	0123
013	3/5

XoXXXX	01478
0147	F 11

XoooXX	0167
016	6/7

oXXXXX	02469
0258	F 11

oXXoXo	0257
025	3/5

ooXooX	0369
036	3/5

oooXoo	048
04	0/2

XXXXoo	0124	0134	01234
0124	-	-	-
0134	F 14	-	-
01234	F 10	F 10	-

oXXXXo	0247	0358	02479
0247	-	-	-
0358	F 14	-	-
02479	F 10	F 10	-

[30] The reader is reminded that set class 3-11(037), for example, and mset class 037 are not the same thing. The former contains pitch classes with any number of representatives. The latter contains exactly one representative of each pc.

Figure 2.26 D.degree for Ze- and Zoe-relations in profile group 3. The returned values are the degrees of doubling required in the parent classes in order to produce Ze- or Zoe-related multiset classes.

XXoXXX	0126	0157	01267	01268	012678
0126	-	-	-	-	-
0157	F 11	-	-	-	-
01267	F 10	F 10	-	-	-
01268	F 10	F 13	F 10	-	-
012678	F 11	F 11	F 10	F 10	-

XoXXXo	0145	0148	0158	0347	01458	014589
0145	-	-	-	-	-	-
0148	F 12	-	-	-	-	-
0158	F 13	F 14	-	-	-	-
0347	F 11	F 12	F 13	-	-	-
01458	F 10	F 12	F 13	F 13	-	-
014589	F 11	F 11	F 11	F 11	F 10	-

oXoXoX	026	0246	0248	0268	02468	02468T
026	-	-	-	-	-	-
0246	**3/5**	-	-	-	-	-
0248	F 12		-	-	-	-
0268	**6/7**	F 12		-	-	-
02468	F 12	**3/6**	F 12	F 10	-	-
02468T	**1/5**	F 11	F 11	**4/7**	F 10	-

XXXXXo	0125	0135	0237	01235	01245	01348	01358	02347	02357	03458	012345	013458	023457	024579
0125	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0135	F 13	-	-	-	-	-	-	-	-	-	-	-	-	-
0237	F 13	F 13	-	-	-	-	-	-	-	-	-	-	-	-
01235	F 10	F 13	F 13	-	-	-	-	-	-	-	-	-	-	-
01245	F 10	F 13	F 13	F 12	-	-	-	-	-	-	-	-	-	-
01348	F 12	F 13	F 13	F 13	F 11	-	-	-	-	-	-	-	-	-
01358	F 12	F 12	F 12	**5/6**	F 11	F 12	-	-	-	-	-	-	-	-
02347	F 12	F 12	F 12	F 11	F 12	F 12	F 12	-	-	-	-	-	-	-
02357	F 12	F 12	F 12	F 11	**5/6**	F 12	F 12	F 12	-	-	-	-	-	-
03458	F 12	F 12	F 12	F 11	F 12	F 12	F 12	F 12	F 13	-	-	-	-	-
012345	F 11	F 11	F 10	F 11	F 10	F 10	F 10	F 10	F 10	F 10	-	-	-	-
013458	F 11	F 11	F 10	F 10	F 10	F 10	F 10	F 10	F 10	F 10	**4/5**	-	-	-
023457	F 11	F 10	F 10	F 10	F 10	F 10	F 10	F 10	F 10	F 10	F 11	F 11	-	-
024579	F 11	F 10	F 10	F 10	F 10	F 10	F 10	F 10	F 10	F 10	F 11	**4/5**	F 11	-

Figure 2.27 D.degree for Ze- and Zoe-relations in profile group 4 (# of objects: 4,5 x 4,5). The returned values are the degrees of doubling required in the parent classes in order to produce Ze- or Zoe-related multiset classes.

Figure 2.28 Ze- and Zoe-related multiset classes. Z- and Zo-related pairs are ranked by d.degree within each profile. Multiset classes are listed by parent class and mfunction, and the shared (except 0 entry) ICV is in the rightmost column.

Gr.	profile	rel.	multiset class A		multiset class B		d.deg	ICVs	
2	**oooXoo**	Zoe	04	<3,1>	048	<1,1,1>	0/2	3/0, 0, 0, 3, 0, 0	0/0, 0, 0, 3, 0, 0
3	**oXoXoX**	Zoe	026	<2,4,2>	02468T	<2,1,1,1,1,1>	1/5	8/0, 8, 0, 8, 0, 4	1/0, 8, 0, 8, 0, 4
2	**ooXooX**	Zoe	036	<5,2,1>	0369	<3,2,1,1>	3/5	11/0, 0, 12, 0, 0, 5	4/0, 0, 12, 0, 0, 5
2	**oXXoXo**	Zoe	025	<5,1,2>	0257	<3,1,2,1>	3/5	11/0, 5, 2, 0, 10, 0	4/0, 5, 2, 0, 10, 0
2	**XXXooo**	Zoe	013	<2,5,1>	0123	<2,3,1,1>	3/5	11/10, 5, 2, 0, 0, 0	4/10, 5, 2, 0, 0, 0
3	**oXoXoX**	Zoe	026	<2,5,1>	0246	<2,3,1,1>	3/5	11/0, 10, 0, 5, 0, 2	4/0, 10, 0, 5, 0, 2
3	**oXoXoX**	Zoe	0246	<7,1,1,1>	02468	<2,1,1,3,1>	3/6	21/0, 9, 0, 8, 0, 7	4/0, 9, 0, 8, 0, 7
3	**XXXXXo**	Ze	012345	<6,1,1,1,1,1>	013458	<1,2,3,2,1,1>	4/5	15/10, 9, 8, 7, 6, 0	5/10, 9, 8, 7, 6, 0
3	**XXXXXo**	Ze	013458	<1,1,3,1,2,2>	024579	<1,1,6,1,1,1>	4/5	5/6, 9, 8, 7, 10, 0	15/6, 9, 8, 7, 10, 0
3	**oXoXoX**	Zoe	0268	<6,2,1,2>	02468T	<4,1,1,2,1,1>	4/7	17/0, 14, 0, 14, 0, 10	7/0, 14, 0, 14, 0, 10
3	**oXXXXo**	Ze	01245	<2,2,3,1,2>	02357	<1,2,6,1,1>	5/6	6/12, 9, 8, 6, 4, 0	16/12, 9, 8, 6, 4, 0
3	**XXXXXo**	Ze	01235	<2,1,1,1,6>	01358	<2,2,3,1,2>	5/6	16/4, 9, 8, 6, 12, 0	6/4, 9, 8, 6, 12, 0
2	**XoooXX**	Zoe	016	<3,3,3>	0167	<8,1,1,1>	6/7	9/9, 0, 0, 9, 9, 9	28/9, 0, 0, 0, 9, 9
3	**oXoXoX**	Zoe	026	<3,3,3>	0268	<8,1,1,1>	6/7	9/0, 9, 0, 9, 0, 9	28/0, 9, 0, 9, 0, 9

048 is Zoe-related to 0004; they share the ICV 3/000300. Figure 2.28 lists together all found Ze/Zoe-related pairs, sorted by doubling degree.[31]

2.2.7 Z-Relation Within a Single Parent Class

Up to this point, I have examined Z-type relations only between pcmultisets from different parent classes, showing the varying degrees to which any pair of parent classes may be Z-related via their multiset offspring. To compare two Z-related pcmultisets from the same parent class, however, is to address something altogether different. For example, mset classes 0012355 and 0223357 are Z-related, sharing the ICV 2/454240. With this information one may assert that their parent classes, 5-2 (01235) and 5-23 (02357), are Z-related *at a doubling degree of 2*; doubling only once the appropriate pcs in each set produces matching intervallic content. But could either one similarly be "Z-related to itself" after a few precise doublings are made? Can either 01235 or 02357 produce two distinct, yet Z-related, off-sprung mset classes? It turns out that both can. However, it takes a few more doublings. In this section, I examine this **self Z-relation** and attempt to uncover what property or properties will allow a single parent class to produce Z-related offspring. As we will see, several distinct, well-known properties—and possibly some still unknown—make this relation possible.

Not all parent classes are capable of self-Z-symmetry. Inversional symmetry, might at first be thought a prerequisite, but this is not so. See 5-2 (01235) or 5-23 (02357) above. Furthermore, I-symmetrical parent classes tend to require more doubling than those that are not, and some parent classes *cannot* produce Z-related offspring at all.

[31] Results of ongoing calculations into higher numbers of elements and objects can be found at https://wfs.gc.cuny.edu/TRobinson/www/dissertation.htm

The parent class 4-7(0145) is inversionally symmetrical, so while we can create pairs of off-sprung mset classes having identical ICVs, they tend often to be inversions of one another. For example, the mfunctions <1,1,1,2> and <2,1,1,1> applied to 4-7 (0145) both create the same mset class, 00145—note that one mfunction retrogrades to the other. We must make enough doublings, therefore, to create asymmetry in the mfunction. In this case the minimum doubling degree to ensure Z-relation is 5. The mfunctions <4,2,2,1> and <2,1,4,2>, when applied to 4-7(0145), create two distinct, yet Z-related, msetclasses. Another inversionally symmetrical parent class, 4-6 (0127), is not so fortunate. Of course, applying the mfunctions <1,1,2,1> and <2,1,1,1> here can produce inversionally related msetclasses—note that one mfunction retrogrades to the other at the appropriate rotation—but no two mfunctions can produce Z-related offspring from this parent class.[32] What is it that permits the self-Z-relation in 5-2 (01235) and 4-7 (0145), but not in 4-6 (0127) and not, for that matter, in 4-2 (0124), 4-4 (0125), and many others? This section considers two chief means of producing Z-related mset classes from the same parent class: Stephen Soderberg's dual inversion and Richard Cohn's transpositional combination.

Figure 2. Dual Inversion. As discussed in Section 2.2.5, many pairs Z-related pc multisets are related by dual inversion and can be generated using Q-grids. In its tally, Figure 2.21 included the pair 023668 and 002368, both offspring of 5-28 (02368). For a more explicit example, see Figure 2.29. Here selections are made from Q-grid 12, 12; 0, 6 to find self-Z-related offspring of 4-25 (0268).

[32] As of this writing, I have checked this tetrachord up to a doubling degree of 12, and I am convinced that no amount of doubling will produce the desired Z-relation.

Figure 2.29 Q-grid generating the self-Z-relation. The appropriate selections are made from grid 12, 12;0, 6 in order to find 0002226668 and 0000226688.

12, 12; 0, 6

0 | 6	1 | 5	2 | 4	3 | 3
0 | 6	e | 7	t | 8	9 | 9

$P_0 = \{Ø\}$	$P^*_0 = \{Ø\}$
$P_{1a} = \{e, 1, 5, 5\}$	$P^*_{1a} = \{1, e, 7, 7\}$
$P_{1b} = \{1, e\}$	$P^*_{1b} = \{5, 7\}$
$P_{1c} = \{e, 1, 5, 7\}$	$P^*_{1c} = \{e, 1, 5, 7\}$
$P_2 = \{Ø\}$	$P^*_2 = \{Ø\}$
$P_3 = \{Ø\}$	$P^*_3 = \{Ø\}$
Q = {e, e, e, 1, 1, 1, 5, 5, 5, 7}	Q* = {e, e, 1, 1, 5, 5, 7, 7, 7, 7}
msc = **0002226668** *4-25 (0268) <3,3,3,1>*	msc = **0000226688** *4-25 (0268) <4,2,2,2>*

ICV = 9/0, 12, 0, 12, 0, 12

Dual inversion does not, however, explain every pair among the multitude of self-Z-related pairs. Parent class 5-2 (01235), already known to produce Z-related offspring, will serve as an example. Q-grid 12, 12; 0, 6 is the only Q-grid that can accommodate selections comprising 5-2 (01235). On it, there are four ways to select transpositions and four to produce inversions. As it does its job with each of the eight possible selection groups, the grid produces in each case a set class that shares the same ICV. As would be expected, every result is merely another member of 5-2 (01235); this set class has no Z-correspondent. If we were to make multiple passes, however, generating multiset classes, we might find a Z-related pair. Also possible is to start with a known Z-related pair of

offspring from this parent class and to assemble the appropriate selections on the grid. In this case there are many known pairs, however, that cannot be reverse-engineered in this way. For one, the mset classes 01235 <1,1,2,3,2> and 01235 <2,2,1,3,1> share the ICV 5/9, 11, 7, 2, 2, 0, but these particular doublings cannot properly be generated on Q-grid 12, 12; 0, 6, which is the only Q-grid even capable of properly generating this set of objects. A simple example will demonstrate this problem. Figure 2.30a displays the Q-grid in question, the selections available that comprise 5-2 (01235), and a pc clockface displaying the dual inversional mappings projected by the Q grid. Each of the eight possible selections as well as their resulting mappings are T_n/T_nI-related, so this one selection set will suffice. In Figure 2.30b, the same information is given but now pitch-class doublings are attempted. In the selection set, it is clear that such doubling would make for improper selections on the Q-grid. The multiplicities of each pc representative, indicated by extra circles the pc clock face, when mapped to their dual inversions do not properly correspond; e.g., two pc7s ought to map to two pc 11s, not to one. What is more, the pitch-class mappings on these clock faces are 1-to-1. No pc is mapped to two different other pcs. The Z-relation among multisets tends to occur when from a single pitch class, one pc representative inverts around x and another representative inverts around y.

To consider the foregoing is merely to scratch the surface. There are many more self-Z-related (and simply Z-related) multiset classes left unexplained by dual inversions. Take 4-1 (0123) for example. This is a parent class that cannot properly be generated on any Q-grid, except in simple (one-axis) inversion. This shortcoming precludes any dual

inversion among offspring of the set class. Yet 0123 <4,2,2,1> is Z-related to 0123 <2,4,1,2>. In cases like these there is something else at work.

Figure 2.30 Q-grid failure. (a) Set class 5-2 (01235) maps to itself under dual inversion. Clockface displays mappings. (b) Doublings known to produce self-Z-relation are attempted on Q-grid and on clockface with no success.

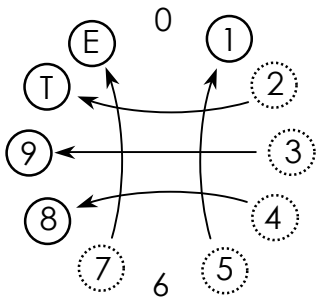
(a)

12, 12; 0, 6

$\begin{bmatrix} 0 & 6 \\ 0 & 6 \end{bmatrix} \begin{bmatrix} 1 & 5 \\ e & 7 \end{bmatrix} \begin{bmatrix} 2 & 4 \\ t & 8 \end{bmatrix} \begin{bmatrix} 3 & 3 \\ 9 & 9 \end{bmatrix}$

$P_0 = \{Ø\}$ $P^*_0 = \{Ø\}$
$P_1 = \{5, 7\}$ $P^*_1 = \{1, e\}$
$P_2 = \{2, 4\}$ $P^*_2 = \{t, 8\}$
$P_3 = \{3\}$ $P^*_3 = \{9\}$

$Q = \{2, 3, 4, 5, 7\}$ $Q^* = \{8, 9, t, e, 1\}$

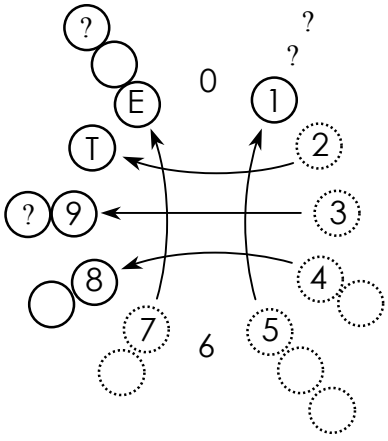


sc = **5-2 (01235)** sc = **5-2 (01235)**

(b)

12, 12; 0, 6

$\begin{bmatrix} 0 & 6 \\ 0 & 6 \end{bmatrix} \begin{bmatrix} 1 & 5 \\ e & 7 \end{bmatrix} \begin{bmatrix} 2 & 4 \\ t & 8 \end{bmatrix} \begin{bmatrix} 3 & 3 \\ 9 & 9 \end{bmatrix}$



<u>Known Z-relation:</u>

{2,3,4,4,5,5,5,7,7} {8,8,9,9,t,e,e,e,1}
msc = **012233355** msc = **001123335**

5-2 (01235) <1,1,2,3,2 > *5-2 (01235) <2,2,1,3,1 >*

<u>Attempted dual inversion:</u>

$P_0 = \{Ø\}$ $P^*_0 = \{Ø\}$
$P_{1a} = \{5, 7\}$ $P^*_{1a} = \{1, e\}$
$P_{1b} = \{5, 7\}$ $P^*_{1b} = \{\text{1}, e\}$ (extra 1!)

P_{1c} = {5} (unbalanced y selection!)
P_2 = {2, 4, 4} (unbalanced x selection!)
P_3 = {3} P^*_3 = {9}

(2) Transpositional Combination. Figure 2.31 revisits our problematic parent class, 5-2 (01235), but finds it not to be the result of self-mapping as in dual inversion. In fact, in Figure 2.31a, the set class is generated by the combination of a subset with its transposition. Adding of set class 3-2 (013) to its transposition at T_2 to form 5-2 (01235) is an example of *transpositional combination* and is designated like so: 3-2 * 2.[33] The superset, 5-2 (01235), thus has the "TC-property," and its *factors* are 3-2 and 2.[34] Cohn calls any set class that cannot be broken down, or factored, into transpositionally related subsets a "prime set-class." Transpositional combination frequently works in a compound fashion. For example, the dyad 2-1 (01) may be transposed by T_1, and the combined result may then be transposed by T_3. This is expressed like so: 1 * 1 * 3 = 6-1 (012345). (Dyads are given simply by interval class; Cohn uses Forte names only for cardinalities 3 and higher.) Set class 6-1 is said by Cohn to be "fully factorable." It can be reduced to exclusively dyadic factors. Of course, this is not the only factorization of 6-1, for 3-1 * 3 is another possibility. The set class in the earlier example above, 5-2 (01235), allows *only* the factorization 3-2 * 2 and cannot be factored exclusively by dyads. It is therefore designated a "semi-factorable" set class. Such set classes may be TC-factorable several different ways, but at least one of the factors will be prime in each. A "semi-prime set class" is one that is TC-factorable, but none of whose factors is a dyad.

---

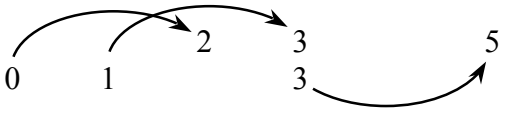
[33] Richard Cohn developed this operation in *Transpositional Combination in Twentieth-Century Music*, Ph.D. dissertation, Eastman School of Music, University of Rochester, 1986. He further explored its analytical application in "Inversional Symmetry and Transpositional Combination in Bartók" *Music Theory Spectrum* 10 (Spring, 1988): 19-42.

[34] "A pitch-class set bears the TC-property if it may be formed by the union of two or more transpositionally related proper subsets whose cardinality equals or exceeds two." Cohn, *Transpositional Combination*, p. 1.

Figure 2.31 Transpositional Combination. (a) Sc 5-12 (01235) shown as the result of TC (pc): 3-1 * 2. (b) Msc shown as the result of TC (pc-rep.): 013 * 2. (c) Z-related mscs shown as the result of TC with varied doubling of factors.

(a)

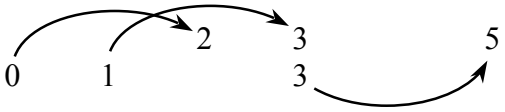
pitch classes:
3-1 * 2



SET CLASS with TC-property: 5-2 (01235)

(b)

pitch-class representatives:
013 * 2



MULTISET CLASS with TC-property: 012335

©

pitch-class representatives:
013 + 235 + 235

T_2 013 = 2 3 5
T_2 013 = 2 3 5
013 = 0 1 3

UNION = 012233355; ICV = 5/9, 11, 7, 2, 2, 0

Z-related

pitch-class representatives:
013 + 013 + 235

T_2 013 = 2 3 5
013 = 0 1 3
013 = 0 1 3

UNION = 001123335; ICV = 5/9, 11, 7, 2, 2, 0

For example, the only TC-factorization of semi-prime 6-15 (012458) is 3-3 * 3-3.[35] These distinctions are important, for, as will be noted below, only those set classes with a particular factorization-type enable the self-Z-relation.

In Figure 2.31a, transpositional combination is dealing in *pitch classes* and, as a result, allows a single pc 3 to stand in for both the initial pc 3 (transposed *to* pc 5) and the resulting pc 3 (transposed *from* pc1).[36] Dealing now in *pitch-class representatives*, the same transpositional combination in Figure 2.31b accounts for both pc3s, generating mset class 012335. Now we have not a *set class* with the TC-property (3-1 * 2 = 5-1 (01235)), but a *multiset class* exhibiting it (013 * 2 = 012335). To achieve self-Z-relation, one needs, as in Figure 2.31c, only to combine this superset 012335 first with one subset, 013, then, *instead*, with the other, 235. The two resulting unions, 012233355 and 001123335, are Z-related. Because the parent class (5-2 (01235)) is at least semi-factorable, one can break it into exactly two T-related components (013 and 235), then add them to one another, doubling one first then the other ((013 + 235) + 013 and (013 + 235) + 235). The resulting multiset classes will be Z-related. The relation will hold even through unlimited doubling of the subsets; one may initially repeat the first component *x* times and the second, *y* times, provided that this is followed by one's repeating the first component *y* times, and the second, *x* times. This may be surprising at first, given the usual focus on inversion when it comes to matters Z-related, but the reader can, to his or her satisfaction, take any semi- or fully factorable set class and perform the steps above. Take, for an easy example, 6-Z19 (013478). This set class is composed of a major (or

---

[35] Transpositional combination of the operands 3-3 and 3-3 is for Cohn a "complex operation,"one in which the cardinality of more than one operand is greater than 2." Cohn, *Transpositional Combination*, p. 85. To solve, add $T_0(014) + T_1(014) + T_4(014)$ to generate 012458.

[36] Cohn notes that TC does not account for pc duplication, but suggests how it could. See *Transpositional Combination*, p. 109.

minor) triad combined with itself at T_1: 037+148. The operation is 3-11 * 1. Combine {0,3,7}, {0,3,7} again, and {1,4,8} to generate {0,0,1,3,3,4,7,7,8}. Then compare this result to the combination of {0,3,7}, {1,4,8}, and {1,4,8}: {0,1,1,3,4,4,7,8,8}. The two results are Z-related. This practice of combining *x* subsets A with *y* subsets B, then combining *y* subsets X with *x* subsets Y, will be referred to here simply as multi-transpositional combination, or MTC.

The distinction, made above, between the semi- or fully factorable set classes and the prime and semi-prime set classes, is an important one because prime and semi-prime set classes cannot have self-Z-related offspring (at least, not through TC). Only semi- and fully factorable set classes can. Many set classes can be factorable in several different ways, but the factorizations enabling the self-Z-relation are those that break a parent class into exactly two components, one of which must be a dyad.

While not all parent classes with Z-related offspring are semi- or fully factorable—some self-Z-related multiset classes are the result of dual inversion—all semi- or fully factorable parent classes can generate Z-related offspring. This is why the inversional symmetry of parent classes previously was so misleading. All inversionally symmetrical parent classes can be expressed as a combination of two inversionally related subsets, but only if those subsets, themselves, are inversionally symmetrical will the TC property be *guaranteed*; i.e., the subset should be able non-trivially to transpose to its inversion. For example, the non-I-symmetrical {016} added to T_2I{016} equals {01268}, a member of 5-15 (01268), which *is not* semi- or fully TC-factorable.[37] On the other hand, the I-symmetrical {0134} added to T_1I{0134} equals {9t0134}, a member of 6-Z13 (013467), which *is* so factorable—{0134} *transposes* to its inversion {3467}. In

[37] This parent class is free to (and does) produce Z-related offspring through dual inversion, however.

other words, it is the transpositional relation that is of interest here, not the inversional one.

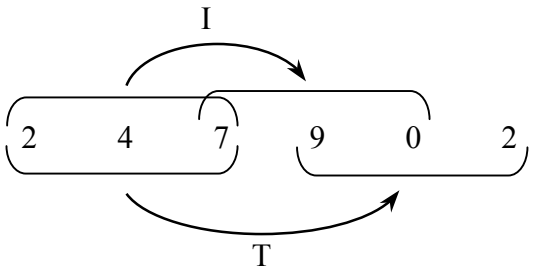
Inversional symmetry in either the sub- or superset is, in fact, somewhat of an impediment to efficient self-Z-relation within a parent class. Where transposition leads toward a Z-relation, inversion tends to lead toward T_n/T_nI-relation. As it turns out, an I-symmetrical parent class composed, like 5-15 (01268) above, of two inversionally related but not inversionally symmetrical subsets may be semi- or fully TC-factorable after all, so long as the T-related and I-related subsets are *different* subset pairs. Furthermore, an I-symmetrical parent class composed, like 6-Z13 (013467) above, of two inversionally related and inversionally symmetrical subsets will be semi- or fully TC-factorable but to generate self-Z-related offspring one may have to use pc duplication *within* the subset being transpositionally combined. Some examples should make these two scenarios perfectly clear.

Parent class 5-35 (02479)—see Figure 2.32a—can be formed by transpositional combination as well as by "inversional combination." That is, to generate this set class, a trichord can be transposed and added to itself or one can be inverted and added to itself, but the generating trichords would be different ones. In Figure 2.32b, multi-transpositional combination of TC-factors will generate the self-Z-relation. Parent class 3-1 (012) on the other hand—see Figure 2.33a—can also be formed by transpositional combination as well as "inversional combination," but its subsets are the same in each case. The dyad that gets transposed is the same one that gets inverted. In Figure 2.33b multi-transpositional comination fails to generate self-Z-related mset classes. The results share an ICV, of course, but they are merely T_n/T_nI-related. In Figure 2.33c, to instill

some asymmetry in the mfunction, the TC operand, 01, receives a pc doubling (001) *before* it is multi-transpositionally combined. After the operation, self-Z-relation is achieved.

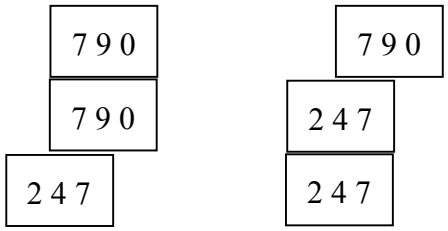
Figure 2.32 Different TC-factors and "IC-factors." In such a case self-Z-relation is easily achieved.

(a)



(b)

Repeated T-factors:



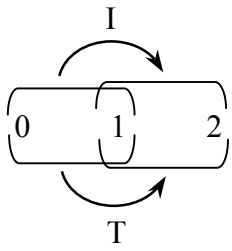
02479 <2,1,1,3,2> 02479 <1,2,2,3,1>
= 02479 <2,2,1,1,3> by inversion/rotation

ICV = 5/0, 9, 7, 2, 13, 0 ICV = 5/0, 9, 7, 2, 13, 0

Z-related

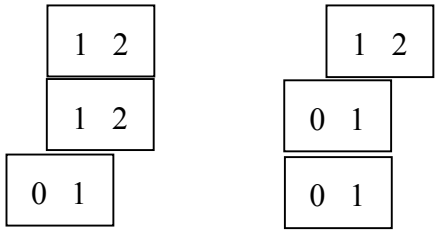
Figure 2.33 Same TC-factors and "IC-factors." In such a case self-Z-relation achieved by using doubling within the subset.

(a)



(b)

Repeated TC-factors:

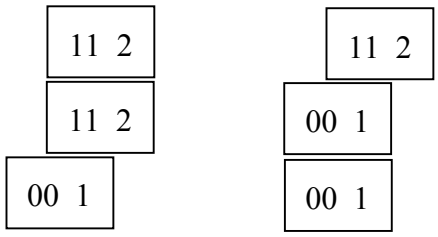


012 <1,3,2> 012 <2,3,1>
= 012 <2,3,1> by inversion

T_n/T_nI-related

©

Repeated TC-factors with doubling:



012 <2,5,2> 012 <4,4,1>

ICV = 12/20, 4, 0, 0, 0, 0

Z-related

At this point we can more generally conclude the following:

If

pcmset A <a, b, ... > = T_n/T_nI pcmset B <n, o, ... >

and

x, y are positive, whole numbers, but $x \neq y$,

pcmset A <x*a, x*b, ... > ∪ pcmset B < y*n, y*o, ... >

will have the same ICV as

pcmset A <y*a, y*b, ... > ∪ pcmset B < x*n, x*o, ... >.

If A is not inversionally symmetrical, the pcmset unions may be Z-related. If A is inversionally symmetrical, the multiplicities mfunction <a, b, … > must be asymmetrical in order for the pcmset unions to be Z-related. Even more generally, the generating pcmsets, A and B, may be Z-related. For example, {1,2,4,8} and {2,4,7,8} are Z-related. They are members of 4-Z29 (0137) and 4-Z15 (0146), respectively. Using multi-transpositional combination we find that

{1,2,4,8} + {2,4,7,8} + {2,4,7,8} = {1,2,2,2,4,4,4,7,7,8,8,8}

= 5-19 (01367) <1,3,3,2,3>; ICV = 10/9, 9, 9, 9, 9, 11

and

{1,2,4,8} + {1,2,4,8} + {2,4,7,8} = {1,1,2,2,2,4,4,4,7,8,8,8}

= 5-19 (01367) <2,3,3,1,3>; ICV = 10/9, 9, 9, 9, 9, 11.

The resulting pcmsets (and msetclasses) are self-Z-related. The initial generating pcmsets may even be Z-related offspring from non-Z-related parents. For example, 4-5

(0126) and 4-16 (0157) are not Z-related. One particular offspring from each, however, form a Z-related pair: 01266 and 01157.[38] After MTC we find that

{3,4,5,9,9} + {8,t,2,2,3} + {8,t,2,2,3} = {2,2,2,2,3,3,3,4,5,8,8,9,9,t,t}

= 7-7 (0123678) <4,3,1,1,2,2,2>; ICV = 11/24, 11, 6, 12, 22, 16

{3,4,5,9,9} + {3,4,5,9,9} + {8,t,2,2,3} = {2,2,3,3,3,4,4,5,5,8,9,9,9,9,t}

= 7-7 (0123678) <2,3,2,2,1,4,1>; ICV = 11/24, 11, 6, 12, 22, 16.

The resulting pcmsets (and msetclasses) are self-Z-related.

Figure 2.34 provides the pair of self-Z-related offspring with the lowest doubling degree for every parent class through cardinality six. The first column lists each parent class followed by its degree of symmetry and its I-symmetry type. The next column indicates whether the parent class is semi- or fully TC-factorable. The next column ("ZTC") indicates (with a "Z") whether the parent set class can be formed by combining a pair of Z-related set classes at some transposition or inversion. As shown above, Z-related *multi*set classes (infinitely many), can be combined to form parent classes, too, thus suggesting self-Z-relation by MTC. Such pairs are not accounted for in this table but must still be kept in mind as a possible rationale for self-Z-related mset classes in some cases. For each parent class, the self-Z-related offspring pair with the lowest doubling degree is given by its mfunction.[39] It is noteworthy that the parent classes exhibiting only EV^0 symmetry are incapable of producing self-Z-related offspring by any means. Future investigation should uncover why this is so.

---

[38] Because no two Z-related pc- (or pcm-) sets can map onto one another either by transposition or inversion, we will not have the problem here that we faced in Figures 2.33a and 2.33b. We may combine through varied repetition *any* member of mset class 01266, whether T_n- or T_nI-type, with *any* member of mset class 01157 and achieved the desired results.

[39] Doubling degrees labeled *F x* record the extent to which the computer program has searched for a self-Z-related match. F = "failed"; x = number of elements in highest mset classes checked so far. It is unlikely for matches to be made at multiplicity this high, but calculations continue.

Figure 2.34 Self-Z-related pairs. Each parent class capable of self-Z-relation is listed with the pair off offspring with the lowest doubling degree. Also indicated is the parent class's symmetry, whether it is semi-/fully factorable, and whether it can be generated by varied repetition of Z-related pc-/pcmultisets.

PARENT CLASS					MFUNCTION PAIR WITH SMALLEST D.DEGREE			
Forte name	symmetry		TC	ZTC	d.deg	Z-related mfunctions		ic-vector
3-1 (012)	1,1	EV^1	ff		6	<4,4,1>	<2,5,2>	12/20, 4, 0, 0, 0, 0
3-2 (013)	1,0				*F 18*			
3-3 (014)	1,0				*F 18*			
3-4 (015)	1,0				*F 25*			
3-5 (016)	1,0				*F 25*			
3-6 (024)	1,1	EV^1	ff		6	<4,4,1>	<2,5,2>	12/0, 20, 0, 4, 0, 0
3-7 (025)	1,0				*F 18*			
3-8 (026)	1,0				*F 18*			
3-9 (027)	1,1	EV^1	ff		6	<4,1,4>	<2,2,5>	12/0, 4, 0, 0, 20, 0
3-10 (036)	1,1	EV^1	ff		6	<4,4,1>	<2,5,2>	12/0, 0, 20, 0, 0, 4
3-11 (037)	1,0				*F 18*			
3-12 (048)	3,3	EV^1	ff		6	<4,4,1>	<5,2,2>	12/0, 0, 0, 24, 0, 0
4-1 (0123)	1,1	ODD	ff		5	<4,2,2,1>	<2,4,1,2>	8/14, 10, 4, 0, 0, 0
4-2 (0124)	1,0				*F 16*			
4-4 (0125)	1,0				*F 16*			
4-5 (0126)	1,0				*F 15*			
4-6 (0127)	1,1	EV^2			*F 16*			
4-3 (0134)	1,1	EV^0	ff		5	<4,2,2,1>	<2,1,4,2>	8/10, 4, 10, 4, 0, 0
4-11 (0135)	1,0				*F 15*			
4-13 (0136)	1,0				*F 15*			
4-Z29 (0137)	1,0				*F 15*			
4-7 (0145)	1,1	ODD	ff		5	<4,2,2,1>	<2,1,4,2>	8/10, 0, 4, 10, 4, 0
4-Z15 (0146)	1,0				*F 17*			
4-18 (0147)	1,0				*F 15*			
4-19 (0148)	1,0				*F 17*			
4-8 (0156)	1,1	EV^0	ff		5	<4,2,2,1>	<2,1,4,2>	8/10, 0, 0, 4, 10, 4
4-16 (0157)	1,0				*F 18*			
4-20 (0158)	1,1	ODD	ff		5	<4,1,2,2>	<2,2,4,1>	8/4, 0, 4, 10, 10, 0
4-9 (0167)	2,2	ODD	ff		6	<3,3,3,1>	<4,2,2,2>	9/12, 0, 0, 0, 12, 12
4-10 (0235)	1,1	ODD	ff		5	<4,2,2,1>	<2,1,4,2>	8/4, 10, 10, 0, 4, 0
4-12 (0236)	1,0				*F 23*			
4-14 (0237)	1,0				*F 18*			
4-21 (0246)	1,1	EV^0	ff		5	<4,2,2,1>	<2,1,4,2>	8/0, 14, 0, 10, 0, 4
4-22 (0247)	1,0				*F 17*			
4-24 (0248)	1,1	EV^2			*F 17*			
4-23 (0257)	1,1	ODD	ff		*5*	<4,2,2,1>	<2,1,4,2>	8/0, 10, 4, 0, 14, 0
4-27 (0258)	1,0				*F 17*			
4-25 (0268)	2,2	EV^0	ff		6	<3,3,3,1>	<4,2,2,2>	9/0, 12, 0, 12, 0, 12
4-17 (0347)	1,1	ODD	ff		5	<4,2,2,1>	<2,1,4,2>	8/4, 0, 10, 10, 4, 0
4-26 (0358)	1,1	EV^0	ff		5	<4,2,2,1>	<2,1,4,2>	8/0, 4, 10, 4, 10, 0
4-28 (0369)	4,4	B	ff		6	<3,3,3,1>	<4,2,2,2>	9/0, 0, 24, 0, 0, 12

Figure 2.34 Self-Z-related pairs (continued)

PARENT CLASS						MFUNCTION PAIR WITH SMALLEST D.DEGREE			
5-1 (01234)	1,1	EV[1]	ff			4	<2,3,1,2,1>	<2,1,2,3,1>	5/13, 9, 7, 2, 0, 0
5-2 (01235)	1,0		sf			4	<1,1,2,3,2>	<2,2,1,3,1>	5/9, 11, 7, 2, 2, 0
5-4 (01236)	1,0					*F 16*			
5-5 (01237)	1,0					*F 14*			
5-3 (01245)	1,0		sf			4	<1,3,2,1,2>	<2,3,1,2,1>	5/11, 4, 7, 7, 2, 0
5-9 (01246)	1,0					*F 16*			
5-Z36 (01247)	1,0					*F 14*			
5-13 (01248)	1,0					*F 14*			
5-6 (01256)	1,0		sf			4	<1,3,2,1,2>	<2,3,1,2,1>	5/11, 2, 2, 7, 7, 2
5-14 (01257)	1,0					*F 14*			
5-Z38 (01258)	1,0					*F 15*			
5-7 (01267)	1,0		sf			2	<1,2,1,2,1>	<1,1,1,2,2>	2/6, 1, 0, 2, 6, 4
5-15 (01268)	1,1	EV[1]				4	<3,1,1,2,2>	<2,1,2,3,1>	5/4, 7, 0, 8, 4, 8
5-10 (01346)	1,0		sf			4	<1,1,3,2,2>	<2,2,3,1,1>	5/7, 7, 11, 2, 2, 2
5-16 (01347)	1,0		sf			4	<1,1,3,2,2>	<2,2,3,1,1>	5/7, 2, 11, 7, 2, 2
5-Z17 (01348)	1,1	EV[1]				*F 16*			
5-Z12 (01356)	1,1	EV[1]				*F 14*			
5-24 (01357)	1,0					*F 14*			
5-27 (01358)	1,0		sf			4	<1,2,1,2,3>	<2,1,2,1,3>	5/2, 4, 7, 7, 11, 0
5-19 (01367)	1,0				Z	1	<1,1,1,2,1>	<2,1,1,1,1>	1/3,1,3,1,3,3
5-29 (01368)	1,0					*F 16*			
5-31 (01369)	1,0					*F 16*			
5-Z18 (01457)	1,0					*F 14*			
5-21 (01458)	1,0		sf			1	<1,1,1,2,1>	<1,2,1,1,1>	1/3, 0, 3, 5, 3, 0
5-30 (01468)	1,0					*F 16*			
5-32 (01469)	1,0		sf			4	<1,2,1,2,3>	<2,1,2,1,3>	5/2, 2, 11, 7, 7, 2
5-22 (01478)	1,1	EV[1]				*F 16*			
5-20 (01568)	1,0		sf			4	<1,3,2,2,1>	<2,3,1,1,2>	5/7, 2, 2, 7, 11, 2
5-8 (02346)	1,1	EV[1]				*F 14*			
5-11 (02347)	1,0					3	<1,1,1,3,2>	<3,1,2,1,1,>	4/4, 4, 7, 5, 4, 0
5-23 (02357)	1,0		sf			4	<1,2,1,3,2>	<2,1,2,3,1>	
5-25 (02358)	1,0		sf			4	<1,1,2,3,2>	<2,2,1,3,1>	5/2, 7, 11, 2, 7, 2
5-28 (02368)	1,0				Z	1	<1,1,1,2,1>	<2,1,1,1,1>	1/1, 3, 3, 3, 1, 3
5-26 (02458)	1,0					*F 16*			
5-33 (02468)	1,1	EV[1]	ff			2	<1,2,1,2,1>	<2,2,1,1,1>	2/0, 8, 0, 7, 0, 4
5-34 (02469)	1,1	EV[1]				*F 15*			
5-35 (02479)	1,1	EV[1]	ff			4	<2,1,1,3,2>	<2,2,1,1,3>	5/0, 9, 7, 2, 13, 0
5-Z37 (03458)	1,1	EV[1]				*F 14*			

Figure 2.34 Self-Z-related pairs (continued)

PARENT CLASS					MFUNCTION PAIR WITH SMALLEST D.DEGREE			
6-1 (012345)	1,1	ODD	ff		6	<2,2,4,1,1,2>	<4,2,2,2,1,1>	9/19, 16, 12, 6, 4, 0
					tie:	<2,1,3,3,2,1>	<2,3,2,3,1,1>	8/22, 18, 11, 5, 2, 0
						<2,4,1,2,1,2>	<4,2,2,1,2,1>	9/18, 15, 10, 10, 4, 0
						<2,1,2,3,3,1>	<2,3,3,1,2,1>	8/22, 16, 11, 7, 2, 0
6-2 (012346)	1,0		sf		3	<1,1,2,2,1,2>	<2,2,1,1,2,1>	3/9, 8, 7, 5, 2, 2
6-Z36 (012347)	1,0				7	<1,2,1,2,4,3>	<2,3,1,1,4,2>	11/14, 9, 22, 10, 6, 6
6-Z37 (012348)	1,1	EV^2			*F 14*			
6-Z3 (012356)	1,0		sf		3	<1,2,1,2,1,2>	<2,1,2,1,2,1>	3/8, 7, 7, 4, 5, 2
6-9 (012357)	1,0		sf	Z	2	<1,1,2,1,2,1>	<2,1,1,2,1,1>	2/5, 7, 5, 3, 5, 1
6-Z40 (012358)	1,0				8	<3,1,1,5,2,2>	<3,1,2,2,5,1>	15/9, 18, 21, 8, 18, 2
6-5 (012367)	1,0		sf		3	<1,2,1,2,1,2>	<2,1,2,1,2,1>	3/8, 5, 4, 5, 6, 5
6-Z41 (012368)	1,0		sf		3	<1,1,2,1,3,1>	<3,1,1,1,1,2>	4/5, 6, 4, 7, 5, 5
					tie:	<1,1,2,2,1,2>	<2,2,1,1,2,1>	3/7, 6, 4, 4, 7, 5
6-Z42 (012369)	1,1	ODD			4	<3,1,1,1,2,2>	<2,1,1,2,3,1>	5/5, 4, 15, 4, 4, 8
6-Z38 (012378)	1,1	ODD	ff		4	<1,3,1,1,2,2>	<1,2,2,1,3,1>	5/11, 4, 1, 4, 12, 8
6-Z4 (012456)	1,1	EV^0	ff		6	<2,2,4,1,1,2>	<4,2,2,2,1,1>	9/15, 14, 6, 12, 6, 4
6-Z11 (012457)	1,0				8	<1,1,2,5,3,2>	<1,2,5,2,3,1>	15/18, 18, 21, 8, 9, 2
6-15 (012458)	1,0				2	<1,1,1,2,2,1>	<1,2,1,2,1,1>	2/6, 3, 6, 7, 3, 1
6-Z12 (012467)	1,0			Z	1	<1,1,1,1,2,1>	<2,1,1,1,1,1>	1/4, 4, 2, 3, 4, 3
6-22 (012468)	1,0			Z	1	<1,1,2,1,1,2>	<2,1,1,1,2,1>	2/3, 7, 1, 7, 3, 5
6-Z46 (012469)	1,0		sf		3	<1,2,1,2,2,1>	<2,1,2,1,1,2>	3/4, 7, 7, 6, 7, 2
6-Z17 (012478)	1,0				2	<1,1,1,1,2,2>	<1,2,1,1,1,2>	2/6, 2, 3, 5, 6, 4
6-Z47 (012479)	1,0				7	<1,3,1,4,2,2>	<2,2,1,4,3,1>	11/6, 9, 22, 10, 14, 6
6-Z6 (012567)	1,1	ODD	ff		4	<2,3,1,1,1,2>	<3,2,1,1,2,1>	5/12, 4, 1, 4, 11, 8
6-Z43 (012568)	1,0		sf	Z	1	<1,1,1,1,2,1>	<2,1,1,1,1,1>	1/4, 3, 2, 4, 4, 3
6-Z44 (012569)	1,0		sf		3	<1,1,1,3,1,2>	<1,2,1,1,1,3>	4/5, 1, 7, 12, 6, 1
					tie:	<1,1,2,1,2,2,>	<2,1,1,2,1,2>	3/5, 2, 8, 9, 7, 2
						<1,1,2,2,2,1>	<2,2,1,1,1,2>	3/7, 2, 7, 9, 6, 2
						<1,2,1,3,1,1>	<1,3,1,1,1,2>	4/7, 1, 5, 12, 6, 1
						<1,2,2,1,2,1>	<2,2,1,2,1,1>	3/8, 2, 5, 9, 7, 2
6-18 (012578)	1,0		sf		3	<1,2,1,2,2,1>	<2,1,2,1,1,2>	3/6, 5, 4, 5, 8, 5
6-Z48 (012579)	1,1	EV^2			*F 16*			
6-7 (012678)	2,2	EV^{0-2}	ff		4	<2,3,1,2,1,1>	<3,2,1,1,2,1>	5/12, 4, 0, 4, 12, 8
6-Z10 (013457)	1,0		sf		3	<1,1,1,2,2,2>	<2,2,2,1,1,1>	3/7, 7, 7, 6, 4, 2
6-14 (013458)	1,0		sf	Z	1	<1,1,1,1,2,1>	<1,2,1,1,1,1>	1/4, 3, 4, 5, 4, 0

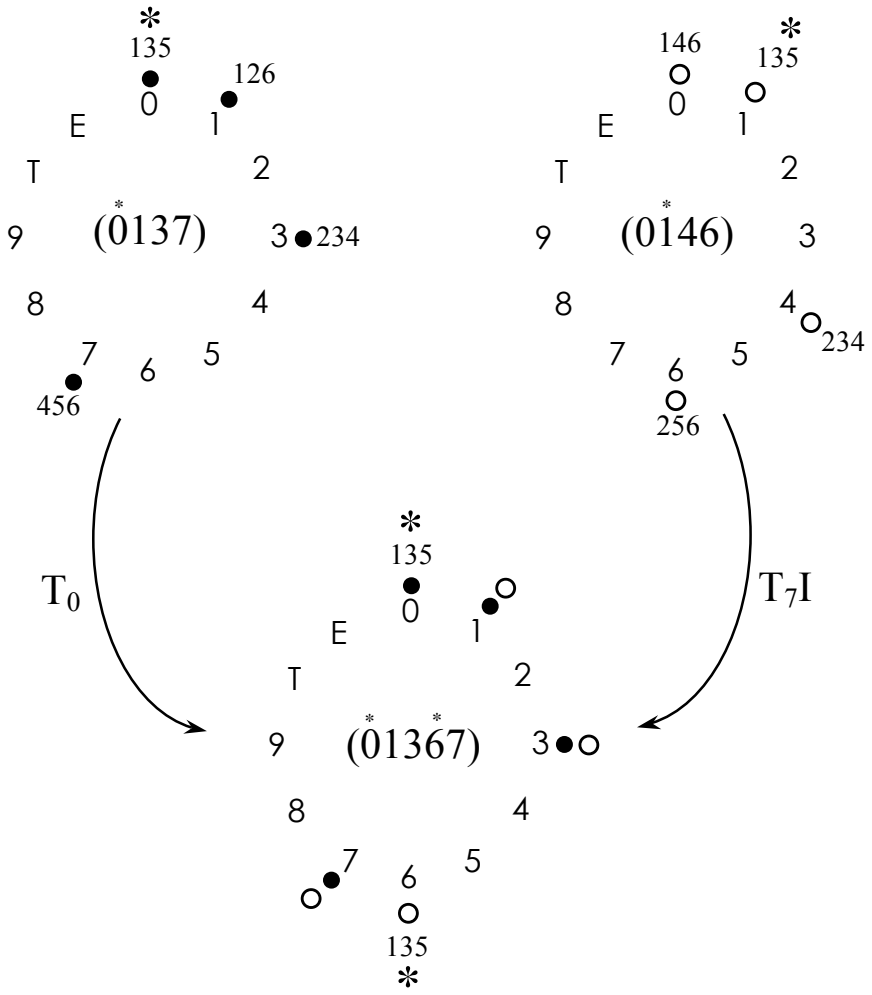
Figure 2.34 Self-Z-related pairs (continued)

PARENT CLASS					MFUNCTION PAIR WITH SMALLEST D.DEGREE			
6-Z13 (013467)	1,1	ODD	ff	Z	4	<3,2,2,1,1,1>	<2,3,1,2,1,1>	5/9, 5, 11, 5, 5, 5
					tie:	<2,3,1,1,1,2>	<3,2,1,1,2,1>	5/9, 4, 8, 4, 7, 8
6-Z24 (013468)	1,0				6	<1,1,2,2,3,3>	<1,2,1,3,3,2>	8/5, 17, 10, 11, 12, 3
6-27 (013469)	1,0		sf	Z	1	<1,1,1,2,1,1>	<1,2,1,1,1,1>	1/3, 3, 6, 3, 3, 2
6-Z19 (013478)	1,0		sf		3	<1,1,1,2,1,3>	<3,1,1,1,1,2>	4/6, 1, 5, 12, 7, 1
					tie:	<1,1,1,3,1,2>	<3,1,1,2,1,1>	4/6, 1, 7, 12, 5, 1
						<1,1,2,1,2,2>	<2,2,1,1,1,2>	3/7, 2, 5, 9, 8, 2
						<1,2,1,2,1,2>	<2,1,2,1,2,1>	3/6, 2, 7, 9, 7, 2
						<1,1,2,2,2,1>	<2,2,1,2,1,1>	3/7, 2, 8, 9, 5, 2
6-Z49 (013479)	1,1	EV^0	ff	Z	4	<2,2,1,1,1,3>	<2,1,3,1,2,1>	5/5, 5, 11, 9, 5, 5
					tie:	<1,2,2,1,3,1>	<1,3,1,1,2,2>	5/4, 7, 8, 9, 4, 8
6-Z25 (013568)	1,0		sf		3	<1,1,1,2,2,2>	<2,2,2,1,1,1>	3/5, 7, 7, 4, 8, 2
6-Z28 (013569)	1,1	EV^2			*F 15*			
6-Z26 (013578)	1,1	EV^0	ff		6	<2,1,2,2,4,1>	<2,2,1,4,2,1>	9/6, 14, 6, 12, 15, 4
6-34 (013579)	1,0				2	<1,1,1,1,2,2>	<1,1,2,1,2,1>	2/1, 8, 3, 7, 3, 4
6-30 (013679)	2,0	(T)	sf		2	<2,1,2,1,1,1>	<2,1,1,1,1,2>	2/3, 3, 9, 3, 3, 5
6-16 (014568)	1,0			Z	2	<1,1,1,2,1,2>	<1,2,2,1,1,1>	2/5, 3, 5, 7, 5, 1
					tie:	<2,1,1,2,1,1>	<2,2,1,1,1,1>	2/6, 2, 3, 7, 6, 2
6-31 (014579)	1,0				2	<1,1,2,1,1,2>	<2,1,1,1,1,2>	2/3, 3, 6, 7, 6, 1
6-20 (014589)	3,3	ODD	ff		3	<3,1,2,1,1,1>	<3,1,1,1,2,1>	4/6, 0, 6, 14, 6, 0
6-Z50 (014679)	1,1	ODD	ff	Z	4	<2,1,2,1,3,1>	<3,1,1,1,2,2>	5/5, 5, 11, 5, 9, 5
						<3,1,1,2,2,1>	<2,2,1,1,3,1>	5/7, 4, 8, 4, 9, 8
6-8 (023457)	1,1	ODD	ff		5	<2,1,3,1,3,1>	<3,3,2,1,1,1>	7/9, 15, 10, 5, 9, 0
					tie:	<2,2,3,1,1,2>	<3,1,2,2,2,1>	6/10, 11, 10, 8, 10, 0
6-Z39 (023458)	1,0				6	<2,3,1,3,2,1>	<3,3,2,2,1,1>	8/12, 17, 10, 11, 5, 3
6-21 (023468)	1,0				2	<1,2,1,1,2,1>	<2,2,1,1,1,1>	2/3, 8, 3, 7, 1, 4
6-Z45 (023469)	1,1	EV^2			*F 16*			
6-Z23 (023568)	1,1	EV^0	ff	Z	4	<2,3,1,2,1,1>	<3,2,2,1,1,1>	5/5, 9, 11, 5, 5, 5
					tie:	<2,3,1,1,1,2>	<3,2,1,1,2,1>	5/4, 9, 8, 7, 4, 8
6-33 (023579)	1,0		sf		3	<1,2,1,2,1,2>	<2,1,2,1,2,1>	3/2, 8, 7, 5, 9, 2
6-Z29 (023679)	1,1	EV^0			4	<2,1,3,1,1,2>	<3,1,2,2,1,1>	5/4, 4, 15, 4, 5, 8
6-32 (024579)	1,1	EV^0	ff		6	<2,2,4,1,1,2>	<4,2,2,2,1,1>	9/4, 15, 10, 10, 18, 0
					tie:	<2,3,2,1,3,1>	<3,3,1,2,2,1>	8/2, 18, 11, 5, 22, 0
						<2,1,2,2,4,1>	<2,2,1,4,2,1>	9/4, 16, 12, 6, 19, 0
						<3,1,1,2,3,2>	<3,2,2,1,3,1>	8/2, 16, 11, 7, 22, 0
6-35 (02468T)	6,6	EV^{0-2}	ff		5	<3,2,2,2,1,1>	<3,2,1,2,1,2>	6/0, 20, 0, 19, 0, 10

The table in Figure 2.34 conceals one important fact but reveals another. Because for any single parent class only the self-Z-related pair with the lowest doubling degree is shown, what remains hidden is the variety of methods a particularly robust parent class frequently has at its disposal to generate so-related offspring. For instance, one set of multiplicities might suggest MTC, while another suggests dual inversion, both within the same parent class. This table shows only the most efficient method. What the results of the calculations reveal, however, is the low doubling degree (1 or 2) some parent classes require for self-Z-relation. What's happening here is not multi-transpositional combination of mset classes. The multiplicity is too low for that to be the case. When the doubling degree is 1, only one pc in the parent class has been doubled at a time. As an example, see 6-Z12 (012467) in the table. The self-Z-related pair is 0124667 and 0012467, as reported by the mfunctions. Why does doubling only the 0 create the same ICV as does doubling only the 6 instead? The answer might lie in the intervallic content of the set classes that are subsets of this parent class.

At the top of Figure 2.35 lies a pair of Z-related set classes: 4-Z29 (0137) and 4-Z15 (0146), each displayed on a pc clock face. The trio of numerals accompanying each element refers to the interval classes measured from just that element to each of the other elements. This is an intervallic "local perspective," so to speak. A specific condition evident in this figure provides for a quick shortcut to self-Z-relation. There are two elements, one from each pcset, that share exactly the same local perspective, 1. These key pcs are marked with a star in the figure. After inverting one pc set against the other—further down in the figure—so that the pc elements with the same local perspective are opposite one another on the clock face (at T_6), the resulting superset is 5-

Figure 2.35 Self-Z-relation through complementary doubling (Z-related subsets). Two elements, one from each of the set classes 4-Z29 (0137) and 4-Z15 (0146), share the same local intervallic perspective, 135. After inverting and combining so that these two elements are T_6 from one another, the union can exhibit self-Z-symmetry through complementary doubling at the two elements. (See listed mfunctions.)



01367 <2,1,1,1,1> = 1/3, 1, 3, 1, 3, 3
01367 <1,1,1,2,1> = 1/3, 1, 3, 1, 3, 3

01367 <3,1,1,1,1> = 3/4, 1, 4, 1, 4, 4
01367 <1,1,1,3,1> = 3/4, 1, 4, 1, 4, 4

01367 <3,1,1,2,1> = 4/5, 1, 5, 1, 5, 7
01367 <2,1,1,3,1> = 4/5, 1, 5, 1, 5, 7

01367 <4,1,1,1,1> = 6/5, 1, 5, 1, 5, 5
01367 <1,1,1,4,1> = 6/5, 1, 5, 1, 5, 5

01367 <4,1,1,2,1> = 7/6, 1, 6, 1, 6, 9
01367 <2,1,1,4,1> = 7/6, 1, 6, 1, 6, 9

19 (01367). This set class now is able to produce self-Z-related offspring simply by adding complementary multiplicities at those specific key pcs. The pairs of mfunctions listed below the superset demonstrate this "complementary doubling." The ICVs are shown to be identical in each pair.[40] This essentially is a streamlined version of multi-transpositional combination. Instead of doubling the entire subsets with the same ICV, one needs only double the single pcs with the same "local perspective."

Not all Z-related pairs share a local intervallic perspective in one of their elements, nor do all pairs of set classes that do are Z-related. Furthermore, the procedure works in some, but not all, scenarios when the set classes in question are not even Z-related, and it fails in others. I hope that further research into this problem will clarify the underlying principles.

Dual inversion, multi-transpositional combination, and the shared local interval perspective all seem to be distinct operations, different means of self-Z-relation. Before ever considering pc duplication, to meaningfully Z-relate a set class to itself would have been illogical, and now continued consideration of pc duplication is beginning to reveal the possibility of a single, overriding concept or principle, one that governs not only self-Z-relation but the Z-relation as well.

2.2.8 Z-, Zo-, Ze-, and Zoe-Related Triples, Quadruples, and More

This exploration of Zo/Ze/Zoe-relations has furthered our understanding of the Z-relation itself, but at the same time it has opened the door for more questions. One not addressed here is the possibility of Z/Zo/Ze/Zoe-related triples, three distinct pcmultisets

[40] The reader will notice that the two subsets share another local intervallic perspective, 234. Performing the procedure, aligning these two pcs by T_6 results in the set class 5-28 (02368). The mfunctions for self-Z-relation of this parent class in Figure 2.34 display the minimal complementary doubling.

that all share the same interval-class vector. A perusal of Figure 2.17, Figure 2.28, or Appendix XX will turn up enough collections to satisfy a modest interest in this phenomenon, which nonexistent among mere non-multisets in 12-tone equal temperament.[41] Naturally such triples, or quadruples, require more and more doubling. Much like one can find a lowest common denominator among several diverse fractions, must there not be one master ICV for each interval profile? Is there one colossal vector from which a hierarchy of multisets spills, each level with less and less multiplicity, all the way down to the individual parent classes that share a profile?

Perhaps we cannot come fully to grips with the Zo/Ze/Zoe-relations before we fully understand the Z-relation itself. What gives rise to it in the first place? By what means can we transform a set into its Z-correspondent. What conditions must be present? After surveying mathematical approaches to homometric (Z-related) sets Clifton Callender and Rachel Hall found an answer to this "Z-relation problem" in Joseph Rosenblatt's use of "spectral units."[42] It is still an open question whether their research is applicable to Z-, or even Ze-, related pcmultisets and whether the results would help us to make better sense of pitch-class doubling in musical situations.

[41] Lewin, "On Extended Z-Triples."
[42] Clifton Callender and Rachel Hall, "Crystallography and the Structure of Z-related Sets" Society for Music Theory Annual Meeting, Nashville, TN, November, 2008.

2.3 Complementation

> The set of 12 pitch-class integers comprises the universal set U, the set of all elements from which sets of cardinal number less than 12 are drawn. The selection of a set of n elements from U effectively divides or partitions U into two sets: the set of n elements selected and the set of 12-n elements not selected. If, for example, M designates a set of 4 elements selected and N designates the remaining set of 8 elements not selected, then M is said to be the *complement* of N with respect to U and N is the complement of M with respect to U.[43]
>
> Given the set X, Y is the complement of X if Y contains all elements of the universe (U) not in X.[44]
>
> For any set, the pitch classes it excludes constitute its *complement*.[45]

Complementation among sets, and abstract complementation among set classes, is well defined in the music-theoretical literature. The definitions usually cite both aggregate completion and the mutual exclusion of pitch classes as necessary conditions. When seeking complementation among pc multisets and mset classes, one need not disrupt these definitions and conditions. Robert Morris, for example, has partitioned the double aggregate—a 24-element collection containing two representatives of each pc—into *dmosaics*.[46] While these structures are most effective when there are three or more parts in the mosaic, when there are only two parts, say {0,1,1,2,3,4} and {0,2,3,4,5,5,6,6,7,7,8,8,9,9,t,t,e,e}, the parts form a complementary pair. To find a complement of a pcmultiset *x*, then, one simply builds a set *y* containing the remaining pitch classes of an *n*-tuple aggregate, where *n* = the highest pc multiplicity in pcmultiset

---

[43] Forte, *Structure*, p. 73-74. (Original emphasis.)
[44] Robert Morris, *Class Notes for Atonal Music Theory* (Lebanon, N.H.: Frog Peak Music, 1991): 5.
[45] Straus, *Introduction*, p. 93. (Original emphasis.)
[46] Morris, "Pitch-Class Duplication," 96-119. Not only does Morris offer suggestions for the enumeration of multisets, but he also applies dmosaics to group theory, orbits, and the Z-relation, which might have some connections to Soderberg's dual inversion, while being more comprehensive. His work in the article, however, is restricted to the double aggregate and, therefore, cannot consider pc multiplicities of three or more.

x. This conception of the complement is merely cumbersome for a pcmset like {0,3,3,7,7}, whose (2-aggregate) complement is {0,1,1,2,2,4,4,5,5,6,6,8,8,9,9,t,t,e,e}, but it is positively useless for the complement of {0,0,0,1,1,4}, a pcmset with thirty elements. Sure, it completes the triple aggregate, but the overwhelming imbalance in size does not reveal much about the intervallic or other relationships between the two. One can see why using Morris's dmosaics of several parts is a more productive way to partition multiple aggregates, but one also might be left longing for another way to match a pcmultiset with a single complementary partner, one much closer in size.

The first step toward finding pc multiset complements of manageable size is to keep in mind that other well-known hallmark of set complements:

> …[F]or it is the extraordinary property of any hexachord that its total intervallic structure is duplicated only by its complementary hexachord, thus equating in these terms the two disjunct halves of any twelve-tone set, and furnishing a unique basis of intervallic identity between two otherwise apparently highly dissimilar pitch complexes.[47]

> To be more specific, the arithmetic difference of corresponding vector entries for complement-related sets is the same as the difference d of the cardinal numbers of the sets, with the exception of the entry for ic6, in which case d must be divided by 2 (since 6 is its own inverse (mod 12)) … In view of this intervallic proportionality it seems reasonable to regard the complement of a set as a reduced or enlarged replica of that set.[48]

Now, the above is true for all complementary (non-multi-) set classes, but it is also true for complementary mset classes, as long as the *n*-tuple aggregate is completed. One can easily find, however, a pair of pcmsets or mset classes for which the above is nearly true

---

[47] Milton Babbitt, "Remarks on the Recent Stravinsky," *Perspectives of New Music* 2/2 (1964): 35-55, reprinted in *The Collected Essays of Milton Babbitt*, ed. Stephen Peles, Stephen Dembski, Andrew Mead, Joseph N. Straus, (Princeton, NJ: Princeton University Press, 2003), 147-171.
[48] Forte, *Structure*, pp. 77-78.

and which are not complements of one another. For example, mset class 0012 has the ICV 1/320000 and mset class 01223467 has the ICV 1/764442. Only the multiplicity in the unison column does not differ by four (or by two in the case of ic6). These certainly are not proper, aggregate-completing complements, but one should not totally write off the apparent affinity.

In my opinion, it is indeed "reasonable to regard" a complement as an "enlarged replica," but it would be another thing altogether to regard any enlarged replica as a complement. This section presents three possible ways to regard complements of pc multisets without bearing the bulk of multiple aggregates. In one method, Milton Babbitt's weighted aggregates act a proxy for the aggregate. In another, Ian Quinn's Fourier-based affinities exhibit what traditional aggregate complements do, balance, but without aggregate completion. In the last, *negative multiplicity* of one or more pitch classes in the complement provide for 1-aggregate (or even 0-aggregate) completion.

2.3.1 Babbitt's Weighted Aggregates

If a two-part partition of the aggregate constitutes a complementary pair, then let a two-part partition of a weighted aggregate constitute a **weighted complementary pair**. It does not require a great leap to partition weighted aggregates but before visiting these structures, it should be made clear one weakness of the concept just proposed in the preceding sentence. Traditional complements partition *the* aggregate; weighted complements partition *a* weighted aggregate. The definite and indefinite articles point out that, in traditional complementation, any pc set or set class has one and only one complement. In weighted complementation, the pc set or set class has any number of

potential complements depending on which weighted aggregate is to be formed in union. This plurality of possibilities notwithstanding, weighted aggregates give us a way to deal with doubled pc representatives. There are even reasonable ways to limit our choices, even to find the "best" fit for complement.

A weighted aggregate is the union of the *parts* of an aggregate, any or all of which have been transposed or inverted such that there is duplication of one or more pcs in the result.[49] For our activities in 12-tone equal temperament, all weighted aggregates will have twelve elements. Take, for example, the following aggregate parts, or *mosaic*: {{01235}{46789te}}.[50] The inversion at T_5I of the first part—{0,2,3,4,5}—combines with the second part to form {0,2,3,4,4,5,6,7,8,9,t,e}, a weighted aggregate. This weighted aggregate may now be two-partitioned into **weighted complements**, say {0,2,3,4,4} and {5,6,7,8,9,t,e}. The preceding is a straightforward process, but if first given the pcmset {0,2,3,4,4} and then asked to find its weighted complement, the pcmset {5,6,7,8,9,t,e} would be only one of several possible answers, for there are any number of weighted aggregates possibly to form. As it happens, any twelve-element pcmultiset can be seen as a weighted aggregate. Pc multiset {0,0,0,0,0,0,2,2,2,2,2,2}, for instance, can be seen as the result of properly transposing or inverting the six parts of the mosaic {{02}{13}{46}{57}{8t}{9e}}. For this reason, to treat a 12-element set as a weighted aggregate and not simply a very big multiset, is analytically meaningless since any and every dodecachord is one. Unless, that is, it is so considered with respect to some

---

[49] Babbitt discusses weighted aggregates in "Set Structure," but the editors of the collection in which the article reappears offer, by way of a footnote, a helpful clarification and example. (See Peles, et al, *Collected Essays*, p. 107n27.)

[50] For more on mosaics see Donald Martino, "The Source Set and Its Aggregate Formations" *Journal of Music Theory* 5/2 (Winter, 1961): 224-73. For partitions, parts, and mosaics as related to the double aggregate, see Morris, "Pitch-Class Duplication," p. 103.

particular preexisting partition. In the context of an array, for example, where aggregate completion is predetermined by multiple partitions, the weighted aggregate can be seen as a transformation of one aggregate in the array.[51] To chop a weighted aggregate in two, then, would seem further to strip the weighted aggregate of its meaning. Because any 12-element pcmset can "be" a weighted aggregate, any x-element pcmset can take any (12-x)-element pcmset as its weighted complement! What apparently is needed is a set of rules for deciding which of the many possibilities is a worthy choice for weighted complement of any pcmset.

It is important to remember that the weighted aggregate is used as a proxy for the aggregate not because of the particular partition that may have produced it, but simply because it is a 12-element pcmset that has at least one doubled pc. So, while I may disregard the weighted aggregate's origins in partitions and arrays, in weighted complementation I focus upon its two other important features: its twelve elements and its pc duplication. Because weighted aggregates contain twelve elements, successful weighted complementation will provide 12-x tones for an x-element pcmset. For example, tetrachord 0013 will have an eight-element weighted complement. Hexachord 000225 will have a six-element weighted complement. This way, the summing to twelve exactly models the summing in traditional complementation. Additionally, in weighted complementation, we will seek a complement that exhibits similar doubling. The weighted complement of tetrachord 0013 not only should have eight elements but also should, like 0013, have one of its pcs doubled; it will be a 7-object octachord. Ideally, when seeking the weighted complement of a pcmset with *x* elements and *y* objects, we

[51] Again, see Morris, "Pitch-Class Duplication," p. 100 for a musical example by Richard Swift, where the surface of the music exhibits the weighted aggregate transformed from the larger, underlying structure.

would select one with 12-*x* elements and $(12 - x) - (x - y)$ objects, but this is not always possible. Lastly, the parent class of pcmset *x*'s weighted complement should be a subset of the (traditional) complement of the pcmset *x*'s parent class. For example, the weighted complement of 0013 (parent class: 3-2 (013)) should be an offspring of a subset of 9-2 (012345679). This is another overture toward traditional complementation; if 3-2 and 9-2 are traditional complements, then weighted complements ought to be found among the subsets of their offspring. Let us now list the criteria and find the weighted complements of a few test cases.

Figure 2.36 displays the preceding three criteria with an added fourth, a tiebreaker that makes reference to the IC vector. It is yet another link to traditional complementation. As we will see, in some cases, weighted complementary pairs can display the same proportion in their interval-class tallies as traditional complements do.

Figure 2.36 Criteria for finding weighted complements

For pcmset M, with *x* elements and *y* objects, to have weighted complement M':

(1) M' must have $12 - x$ elements,
(2) M' must have $(12 - x) - (x - y)$ objects (only if possible), and
(3) The parent class of M' must be a subset of the complement of the parent class of M'.

Of the remaining candidates for M':

(4) M' will be the closest to a "reduced or enlarged replica of" M.

Test Case #1: Complement of 03358

According to criteria (1) and (2), the weighted complement of 03358 ought to have seven elements and six objects. This is a hexachordal parent class with one pc duplicated.

Keeping criterion (4) in mind, one need only scan Appendix 1, the table of mset classes, to find a 6-object septachord whose ICV is an enlarged replica of that of 03358. Because the ICV of 03358 is 1/023130, we would expect, as attested by Forte above, that its complement's ICV has exactly 2 more entries in each position of its vector, except in the ic6-position, where there should only be one more. Adding 0/222221 to 1/023130, we have 1/245351. As it turns out, there is a pcmset with this vector: 0124479. Its parent class, 6-Z47 (012479), is a subset, as requested by criterion (3), of the parent class of 03358, 4-26 (0358). The weighted complement of 03358 is 0124479.

Test Case #2: Complement of 0112

This 3-object tetrachord ought to have as its weighted complement a 7-object octachord. Its ICV is 1/410000, so we would seek an octachord with the ICV 1/854442; this time each position in the vector is increased by four, except ic6 (by 2) and ic0 (by 0 according to criterion (2)). Unfortunately, there is no such octachord. With a little sketching, one can easily conclude that this ICV sought is, indeed, impossible; the eight ic1s and one ic0 may be present simultaneously, but they preclude the presence of two ic6s. Scanning the eight-element offspring of the septachordal parent classes in Appendix 1, however, one can find two ICVs that are minimal variations of our ideal, 1/854442. Since a total of 28 intervals in the vector must be maintained, the "closest" another vector can get is to register one fewer of one ic and one more of another. The two mset classes that do so are 01223467 (ICV: 1/764442) and 01223567 (ICV: 1/754452). (Compare with 1/854442.) Both of these satisfy criteria (1) – (3) and, as best as possible, criterion (4). One can further test these two candidates, though, by taking theirs ICVs, *subtracting* four from

each ic entry (two from ic6, none from ic0) to determine whether either of these candidates has some, more perfect, weighted complement out there that is *not* 0112. ICV 1/764442 – 0/444442 = 1/320000, which happens to be the ICV of 0012, a different tetrachord. It seems this candidate is already taken. If 01223467 is unavailable, what about 01223567? Its ICV is 1/754452, and after subtracting 0/444442 we're left with ICV 1/310010. This is an impossible vector. So, of the two candidates 01223567, apparently unattached, seems the better weighted complement of 0112.

Test Case #3: Complement of 000133557

If Test Case #2 raised some doubts, Test Case #3 will only further them. The ICV of 000133557 is 5/386491. Its weighted complement ought to be a trichord (3 elements), but according to criterion (2) it ought also to have -1 objects! The only available avenue in this case is to find a three-element mset class—with 1, 2, or 3 objects—whose ICV somehow resembles 5/386491. One option would take away one from each ic entry equally until only a total of three entries are left. (Trichordal ICVs have three entries.) The result is ICV 0/010020, which happens to be the vector of mset class 027. Of course, any number of nonachordal IC vectors could be reduced in the similarly down to 027. Furthermore, 027 has three objects and three elements; it has no doubling. It can be treated like the set class 3-9 (027) and, not needing a weighted complement, simply be paired with 9-9 (01235678t). Another option would take away entries in a similar fashion but leave one entry in the ic0 column, thus ensuring pc duplication. The result would be 1/000020. This is the ICV for 005. Once again, however, any number of ICVs could reduce to this same result.

The preceding test cases ultimately are unsatisfying. It seems that when starting with a pcmset or mset class with fewer than six elements, finding the weighted complement is rather straightforward. The reverse, however, is more problematic. Hexachords, whatever the number of their objects, turn out to be self-complementary in this practice. Some pcmsets or mset classes can be weighted complements to two or more others that, themselves, are Z-related to one another. Moreover, the criteria I have proposed are quite arbitrary. Why must the doubling in each complement be the same? And if they must not, how exactly should the ic0 entry differ? The non-commutativity of weighted complementation is perhaps its biggest weakness. Coupling one mset class with two or more weighted complements would be acceptable if it were exclusive, i.e., if these complements didn't have other complements weighting in the wings.

One important role that emerges here, however, is that of the ICV as chief indicator of potential complementation. Perhaps the criterion requiring a sum of 12 elements is what may be needlessly inhibiting our results. Mapping successful test case (#1) above on the pc clockface reveals something important: complementation forms a balanced arrangement in pc space. The next section uses a measurement of pitch-class balance as a surrogate for the important ICV and couples pcmsets of various cardinalities, achieving a useful pseudo-complementation.

2.3.2 Quinn's Fourier Balances

One recent theory of chord quality assesses pc collections according to how much (or how little) they exhibit five specific properties. Any given collection may be assigned five values, each reflecting the collection's affinity with one of five prototypical

collections. Ian Quinn's theory of "General Equal-Tempered Harmony" takes David Lewin's five *Fourier properties* and, applying each to a metaphorical scale (the *Fourier balance*), creates an elegant tool for quantifying exactly how much a pc collection exhibits these properties.[52] In this section, I use one of Quinn's Fourier balances not to "weigh" individual collections on the scale, but to find pairs of them that, when measured on it, "balance" one another. These pairs are considered complements of a sort, in that together they achieve balance in a particular domain.

A Fourier balance is a way to measure the degree to which a pc collection (or set class) exhibits a particular Fourier property. A chord has Lewin's Fourier property 6 (FOURPROP(6) or "Whole-tone-scale property") if it "has the same number of notes in one whole-tone set, as it has in the other."[53] Pc set {0, 1, 4, 5}, then, has this property, while the whole-tone scale, itself, definitely does not. In fact, the whole-tone scale has six tones from one whole-tone scale and none from the other. One might say that the *less* a collection exhibits the whole-tone-scale property, the *more* it resembles a whole-tone scale. By constructing a two-armed scale, or balance, placing the tones from one whole-tone scale on one side, and the tones from the other on the other side, Quinn helps us visualize how collections that have this property are "balanced" on the scale, while those that don't will "tip" the scale. Figure 2.37 reproduces Quinn's Figure 3.2, where we see Fourier balance 6. One can easily envision my earlier example, {0, 1, 4, 5}, represented on the balance, two pcs on one side (0 and 4) and two on the other (1 and 5). If each pc

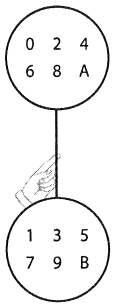
---

[52] In his "General Equal-Tempered Harmony, Part One" *Perspectives of New Music* 44/2 (Summer, 2006): 114-158 and "General Equal-Tempered Harmony, Part Two" *Perspectives of New Music* 45/1 (Winter 2007), Quinn shies away from the term "quality," which he used freely in his dissertation, the work in which the articles originated. See "A Unified Theory of Chord Quality in Equal Temperaments," Ph.D. dissertation, Eastman School of Music, University of Rochester, 2004.

[53] Quinn quotes Lewin in "Unified Theory," p. 78.

"weighed" one "unit", the collection would be balanced on the scale. Any collection that has Fourier property 6 would be so balanced on Fourier balance 6. All six tones of one whole-tone collection, however, if "placed" on the balance would tip it dramatically to one side. Thus, the balance can measure the extent to which any chord exhibits the whole-tone-scale property. The stronger one tips the scale, the more it exhibits the property.

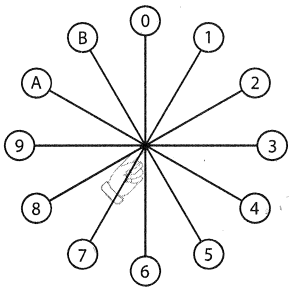
Figure 2.37 Quinn's Fourier Balance 6 *(from Quinn, "Unified Theory," Figure 3.2)*



Quinn constructed one balance for each of Lewin's properties. The additional ones, Fourier balances 4, 3, 2, and 1, measure respectively, the diminished-seventh-chord property, the augmented-triad property, the tritone property, and the exceptional property. In Quinn's theory, any chord or collection is measured with regard to all five properties simultaneously, producing a kind of profile for every chord. For our study of complementation, we will use only Fourier balance 1, the balance that measures the degree to which a chord exhibits Lewin's "exceptional property." Quinn quotes Lewin: "A chord has this property if it 'can be expressed as a disjoint union of tritone sets and/or

augmented-triad sets."[54] The Fourier balance F(12,1), modeled after this property (see Figure 2.38), resembles the familiar pitch-class clockface.[55] More so than others, this particular Fourier balance has applications in voice leading—a move on the balance from one pan to an adjacent one represents a move of a semitone in pc space[56]—and will serve to demonstrate an alternate form of pseudo-complementation among pc multisets.

Figure 2.38 Quinn's Fourier Balance 1 *(from Quinn, "Unified Theory," Figure 3.7)*

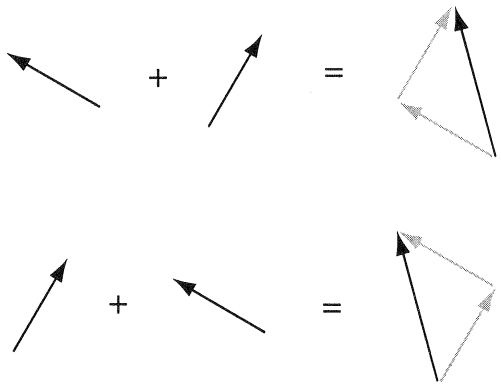


To measure the amount of force a particular collection exerts upon and "tips" a Fourier balance, Quinn uses simple vector addition: "In elementary physics, forces are represented as arrows that can be added to one another in the manner depicted in Figure 3.10 [see my Figure 2.39] — to add two arrows, move them relative to each other (taking care not to rotate them) so that the tail of one is attached to the head of another; then draw a new arrow from the free tail to the free head. The new arrow is the sum of the original

[54] Quinn, "Unified Theory," p. 83. The reader may wonder why there is neither a Fourier property 5 nor a Fourier balance 5. Quinn constructs a Fourier balance 5 and demonstrates its functional equivalence (and relation by multiplication) to Fourier balance 1. He writes, "It is presumably the fact that this one property corresponds to two structurally distinct balances that caused Lewin to name it the 'exceptional property' in 1959." ("Unified Theory," p. 85.)
[55] Quinn uses the shorthand "F(12,1)" to represent **F**ourier Balance **1** in **12**-tone equal temperament.
[56] Thomas Robinson, "The End of Similarity? Semitonal Offset as Similarity Measure." Paper presented at the annual meeting of the Music Theory Society of New York State, Saratoga Springs, NY, April 2006, shows a correlation between positions in Joseph Straus's voice-leading space and Quinn's F(12,1) measurements.

two." The length of this new arrow represents tipping force or *magnitude*, measured by Quinn in "lewins" (Lw), and the direction of the arrow is the direction of the scale's tipping. When this direction is a whole number, the collection is tipping directly toward a particular pc. Sometimes however, the direction lies somewhere between two pcs. Wherever it falls, this measurement of direction may seem trivial because it is so easily altered by the transposition or inversion of the pc collection. Indeed, while one can measure a set class's magnitude on the balance, one cannot measure its direction, only that of any of its specific pcset members. As we will see below, however, the direction is relevant to complementation, at least with respect to the precise location of the tipping between two pcs.

Figure 2.39 Principle of arrow addition *(from Quinn, "Unified Theory," Figure 3.10)*



It should be immediately apparent that semitonal clusters will tip the F(12,1) balance the most. It should also be clear that certain maximally even collections will be perfectly balanced. The whole-tone scale distributes its tones evenly around the balance, as do the diminished-seventh chord and the total aggregate. After some examination, one

should find some other such balanced collections. Set classes 4-9 (0167) and 4-25 (0268), for example, are balanced, exerting a tipping force of 0.00. (The direction, then, of any of the set classes members is also 0.00; if there is no tipping, there is no tipping direction.) Figure 2.40 lists all of the set classes that are balanced in F(12,1). Each of these exhibits Fourier property 1 and, revisiting Quinn's quoting of Lewin, "can be expressed as a disjoint union of tritone sets and/or augmented-triad sets." Looking at Fourier Balance 1, we can see why this is so. The two tones of a tritone lie exactly

Figure 2.40 Balanced Set Classes in F(12,1)

2-6 (06)
3-12 (048)
4-9 (0167)
4-25 (0268)
4-28 (0369)
5-22 (01478)
6-30 (013679)
6-20 (014589)
6-7 (012678)
6-35 (02468T)
7-22 (0125689)
8-28 (0134679t)
8-25 (0124678t)
8-9 (01236789)
9-12 (01245689t)
10-6 (012346789t)
12-1 (0123456789te)

opposite one another on the balance. The augmented triad is similarly balanced. Because the tones, in effect, cancel out each other's force, Quinn calls these "annihilating pairs" and "annihilating triples." The whole tone scale is composed of two such triples ({0, 4, 8} and {2, 6, t}) or three such pairs ({0, 6}, {2, 8}, and {4, t}). In fact, any of the

balanced set classes in Figure 2.40 can be broken down into annihilating pairs and/or triples (discrete 181ritons and/or augmented triads). Conversely, one can create a balanced collection simply by stacking up various 181ritons and augmented triads.

Two pc sets are understood to be complements of one another if they have no pcs in common and if their union is the aggregate. The set classes of which they are members are also understood, then, to be complements, or, more specifically, abstract complements. Because the aggregate is balanced on F(12,1), any pair of complementary pcsets necessarily will exert the same force, but in opposite directions, thus annihilating each other. Quinn writes,

> "Complementation always effects a reversal of the arrow, which is the same as a half-rotation. This somewhat surprising result is easily explained: (1) the aggregate has Fourier Property 1, since it can be partitioned into annihilating subsets; (2) any chord added to its complement forms the aggregate; (3) the force exerted by a chord on Fourier Balance 1 annihilates the force exerted by its complement."[57]

So, by definition, complements must complete the aggregate, but in so doing they also exhibit balance. We will seize upon this latter quality, in order to devise a modified definition of complementation to suit pc multisets.

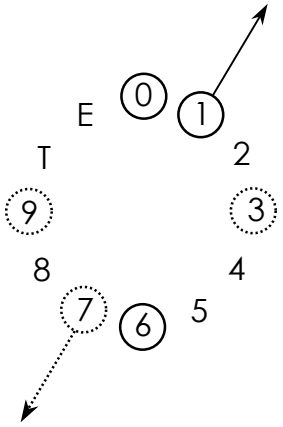
Complementary pairs forming the aggregate are well known, but there are many other pairs that combine to form any of the other balanced set classes in Figure 2.40. We will look for chords that may be combined to form one of these balanced chords, much as chords combine to form the aggregate. For if two chords of equal force (Lw) are situated (transposed and/or inverted) so that their directions are opposite one another on the clockface, the union of the two will exhibit Fourier Property 1, but because they do not complete the aggregate, the two chords cannot properly be called complements.

---

[57] Quinn, "Unified Theory," p. 93-94.

Therefore, they will be called **F1-complements**; their union is balanced in F(12,1).[58] Figure 2.41 displays two set classes, 3-5 (016) and 3-8 (026) on the pc clockface. The arrows representing their Fourier values are of same magnitude and in complementary (180 degrees apart) direction. Just as the aggregate can be broken down into many different complementary pairs, so can this union, 6-30 (013679), be broken down into several different F1-complementary pairs. Figure 2.42 shows two other pairs, both uniting to form the same balanced chord. In each case, it is a simple task to tease out the annihilating pairs.

Figure 2.41 F1-Complementation



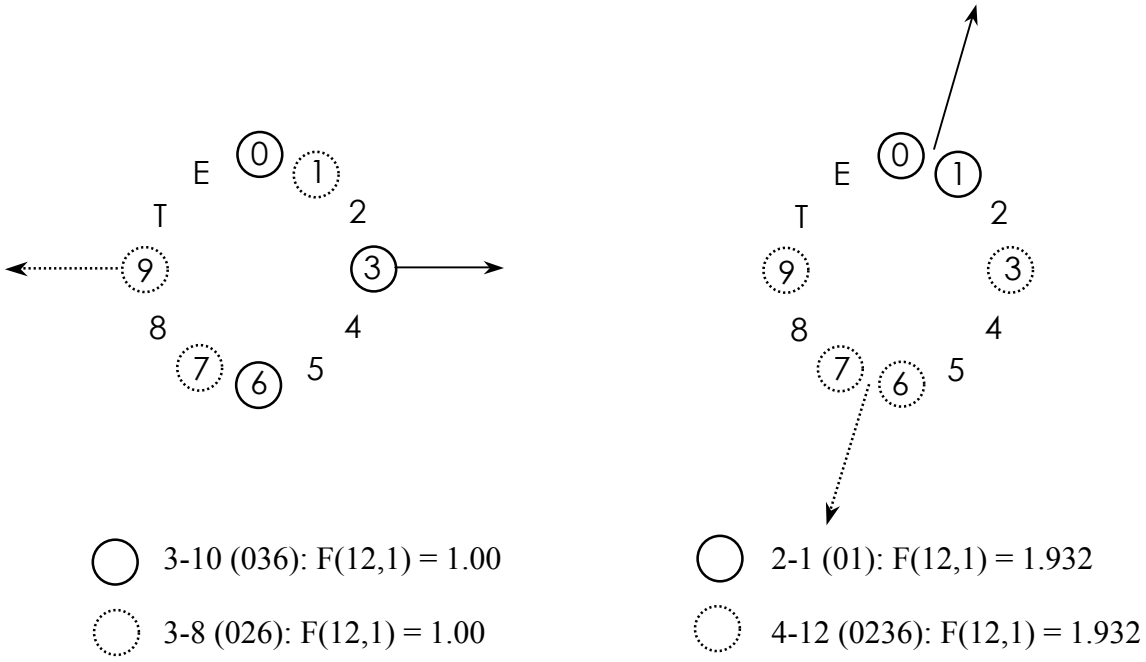
◯ 3-5 (016): F(12,1) = 1.00

◯ 3-8 (026): F(12,1) = 1.00

Union = 6-30 (013679), balanced in F(12,1)

[58] It would be a fallacy to state outright that because any complementation produces balance, any balance thus produces complementation. The term F1-complementation should remind the reader that balance in F(12,1) is the required property. There is not necessarily aggregate completion, but there is necessarily completion of a balanced superset of some kind.

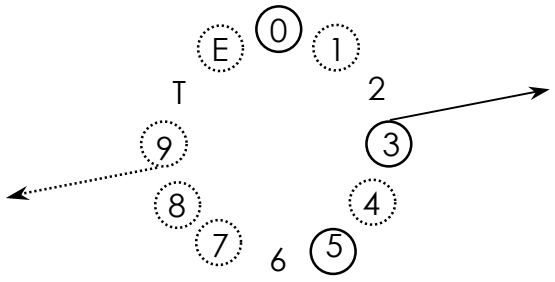
Figure 2.42 Other F1-Complementary pairs. Unions = 6-30 (013679), balanced in F(12,1).



All that is required for a pair of set classes to be F1-complementary, is that the two magnitudes in F(12,1) are identical and that one may be transposed or inverted so that its direction registers exactly six more than the other (mod 12). Matching the magnitudes is simple enough, but some explanation of the directions is in order. Set class 3-2 (013) exerts a force of 2.394 lewins, but we cannot identify a direction until we choose a member of the class, a particular pcset. For example, {0, 1, 3} tips in the direction of 1.29 (slightly closer to pc1 than to pc2), while {4, 5, 7} tips in the direction of 5.29. A transposition of any pcset always changes its direction in F(12,1) by exactly 1.00 for each semitone of transposition. Twelve members of this set class (the T_n-types) will have directions 0.29, 1.29, …, 11.29. The remaining twelve (the T_nI-types), however, will have directions 0.71, 1.71, …, 11.71. Notice that the value to the left of the decimal point changes with transposition, but the value to the right of the decimal

point does not. Only inversion affects this fraction, by subtracting from one. Therefore, when assessing set classes we can look at the direction of its prime form and disregard the whole number value and look only at the post-decimal suffix, for this is the value that must be identical (or they may sum to 1) in two set classes if they are to be F1-complementary. For example, 3-7 (025) and 6-Z46 (012469) have the same magnitude in F(12,1): 1.506 lewins. If their directions can be set exactly six semitones apart, they are F1-complements. Sc 3-7 (025) in prime form has a direction of 2.18, and 6-Z46 (012469) in prime form tips in the direction of 1.82. If the prime form of 6-Z46 is transposed by T_7 to become {7, 8, 9, e, 1, 4}, its direction then will be 8.82. (Simply add 7 to the original direction, 1.82.) The directions of our pcsets now are nearly, but not exactly, complementary: 2.18 and 8.82. One of the two needs to be inverted. If the prime form of 3-7 (025) is inverted by T_5I, its direction will be 2.82, which is complementary to 8.82. Figure 2.43 displays the transpositions and inversions of these set classes as well as the complementary magnitudes and directions of their arrows. All our efforts transposing and inverting would not have been necessary, of course, if we merely noted the post-

Figure 2.43 More F1-Complementation



3-7 (025): F(12,1) = 1.506

6-Z46 (012469): F(12,1) = 1.506

decimal suffixes xx.82 and xx.18 in the directions of the set classes and the fact that they sum to 1. This lets us know that, at some transposition or inversion, the directions are complementary. If, in F(12,1), two set classes have a direction of xx.00 or xx.50 (and their magnitudes are the same) not only are they F1-complements of one another, but even if one of the two is inverted they are still so related.

Figure 2.44 lists the magnitudes, in lewins, for all set classes, from highest (most unbalanced or "chromatic") to lowest (balanced). The traditional aggregate-completing complements are given across from one another. (Partner-less hexachords are self-complementary.) Also listed are the directions as produced by the prime forms.[59] There are only 24 distinct unique magnitudes exhibited by the 218 set classes (including aggregate and null set), and the table displays the clustering of set classes around each magnitude. Each of these clusters represents a single **balance class**.[60] A balance class is a collection of all set classes that share a particular magnitude and that have complementary directions. As can be seen in Figure 2.44 all set classes that share a given magnitude also *do* have complementary directions. There are no two set classes that have the same magnitude yet have incompatible directions, say, 1.50 and 2.63. In every balance class there are some set classes shown in boldface. These are **F1-primes**; they contain no annihilating pairs or triples. Any non-F1-prime set class in a balance class can be reduced to an F1-prime simply by eliminating 185 tritones and or augmented triads. Each prime is a representative, so to speak, of all the members in its balance class, for the non- primes in the class are merely bloated versions of the primes, having 185 tritones or

[59] Variation of +-.01 in the directions is to be expected in the Fourier calculations.
[60] I'd like to thank Joe Straus for suggesting this term.

Figure 2.44 F(12,1) Values for all set classes

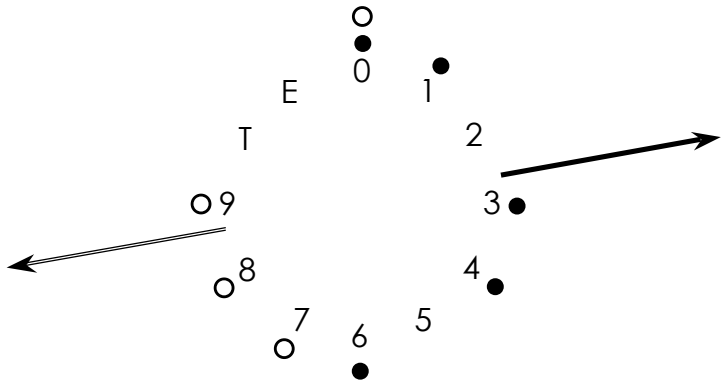
Set Class	Dir.	F(12,1)	Dir.	Set Class
6-1 (012345)	2.50	**3.864**		
5-1 (01234)	2.00	**3.732**	3.00	7-1 (0123456)
4-1 (0123)	1.50	**3.346**	3.50	8-1 (01234567)
6-2 (012346)	2.50	**3.346**		
5-2 (01235)	2.08	**3.235**	2.92	7-2 (0123457)
4-2 (0124)	1.67	**2.909**	3.33	8-2 (01234568)
5-3 (01245)	2.33	**2.909**	2.67	7-3 (0123458)
6-Z36 (012347)	2.33	**2.909**	2.67	6-Z3 (012356)
6-8 (023457)	3.50	**2.828**		
3-1 (012)	1.00	**2.732**	4.00	9-1 (012345678)
4-3 (0134)	2.00	**2.732**	3.00	8-3 (01234569)
5-4 (01236)	2.00	**2.732**	3.00	7-4 (0123467)
5-8 (02346)	3.00	**2.732**	4.00	7-8 (0234568)
6-Z37 (012348)	2.00	**2.732**	3.00	6-Z4 (012456)
4-10 (0235)	2.50	**2.449**	4.50	8-10 (02345679)
6-9 (012357)	2.50	**2.449**		
3-2 (013)	1.29	**2.394**	3.71	9-2 (012345679)
4-4 (0125)	1.71	**2.394**	3.30	8-4 (01234578)
5-10 (01346)	2.70	**2.394**	2.29	7-10 (0123469)
5-5 (01237)	1.71	**2.394**	3.30	7-5 (0123567)
5-9 (01246)	2.29	**2.394**	2.70	7-9 (0123468)
6-Z10 (013457)	3.30	**2.394**	3.30	6-Z39 (023458)
4-11 (0135)	2.11	**2.236**	2.89	8-11 (01234579)
5-11 (02347)	2.89	**2.236**	3.89	7-11 (0134568)
6-Z40 (012358)	2.11	**2.236**	2.89	6-Z11 (012457)
3-6 (024)	2.00	**2.000**	3.00	9-6 (01234568t)
5-Z12 (01356)	3.00	**2.000**	2.00	7-Z12 (0123479)
5-Z36 (01247)	2.00	**2.000**	3.00	7-Z36 (0123568)
6-14 (013458)	3.00	**2.000**		
2-1 (01)	0.50	**1.932**	4.50	10-1 (0123456789)
3-3 (014)	1.50	**1.932**	3.50	9-3 (012345689)
4-12 (0236)	2.50	**1.932**	4.50	8-12 (01345679)
4-5 (0126)	1.50	**1.932**	3.50	8-5 (01234678)
4-7 (0145)	2.50	**1.932**	2.50	8-7 (01234589)
5-13 (01248)	1.50	**1.932**	3.50	7-13 (0124568)
5-16 (01347)	2.50	**1.932**	2.50	7-16 (0123569)
5-6 (01256)	2.50	**1.932**	2.50	7-6 (0123478)
6-15 (012458)	2.50	**1.932**		
6-21 (023468)	3.50	**1.932**		
6-5 (012367)	2.50	**1.932**		
6-Z42 (012369)	1.50	**1.932**	3.50	6-Z13 (013467)
5-23 (02357)	3.23	**1.880**	3.76	7-23 (0234579)
2-2 (02)	1.00	**1.732**	4.00	10-2 (012345678t)
4-13 (0136)	2.00	**1.732**	3.00	8-13 (01234679)
4-21 (0246)	3.00	**1.732**	2.00	8-21 (0123468t)
4-6 (0127)	1.00	**1.732**	4.00	8-6 (01235678)
5-Z17 (01348)	2.00	**1.732**	3.00	7-Z17 (0124569)
5-Z37 (03458)	4.00	**1.732**	4.00	7-Z37 (0134578)
6-Z41 (012368)	2.00	**1.732**	3.00	6-Z12 (012467)
6-Z45 (023469)	3.00	**1.732**	4.00	6-Z23 (023568)
3-7 (025)	2.18	**1.506**	2.83	9-7 (01234578t)
4-14 (0237)	2.18	**1.506**	4.18	8-14 (01245679)
5-14 (01257)	2.18	**1.506**	2.83	7-14 (0123578)
5-24 (01357)	2.83	**1.506**	2.18	7-24 (0123579)
5-25 (02358)	2.83	**1.506**	4.18	7-25 (0234679)
6-Z46 (012469)	1.82	**1.506**	3.17	6-Z24 (013468)

Set Class	Dir.	F(12,1)	Dir.	Set Class
2-3 (03)	1.50	**1.414**	3.50	10-3 (012345679t)
3-4 (015)	1.50	**1.414**	3.50	9-4 (012345789)
4-17 (0347)	3.50	**1.414**	4.50	8-17 (01345689)
4-Z15 (0146)	2.50	**1.414**	2.50	8-Z15 (01234689)
4-Z29 (0137)	1.50	**1.414**	3.50	8-Z29 (01235679)
5-26 (02458)	3.50	**1.414**	3.50	7-26 (0134579)
5-Z18 (01457)	3.50	**1.414**	5.50	7-Z18 (0145679)
5-Z38 (01258)	1.50	**1.414**	3.50	7-Z38 (0124578)
6-16 (014568)	4.50	**1.414**		
6-22 (012468)	2.50	**1.414**		
6-27 (013469)	2.50	**1.414**		
6-Z38 (012378)	1.50	**1.414**	3.50	6-Z6 (012567)
4-22 (0247)	2.80	**1.239**	2.20	8-22 (0123568t)
5-27 (01358)	2.20	**1.239**	2.80	7-27 (0124579)
6-Z47 (012479)	1.20	**1.239**	3.80	6-Z25 (013568)
6-32 (024579)	4.50	**1.035**		
1-1 (0)	0.00	**1.000**	5.00	11-1 (0123456789t)
2-4 (04)	2.00	**1.000**	3.00	10-4 (012345689t)
3-10 (036)	3.00	**1.000**	2.00	9-10 (01234679t)
3-5 (016)	1.00	**1.000**	4.00	9-5 (012346789)
3-8 (026)	2.00	**1.000**	3.00	9-8 (01234678t)
4-18 (0147)	2.00	**1.000**	3.00	8-18 (01235689)
4-19 (0148)	1.00	**1.000**	4.00	8-19 (01245689)
4-24 (0248)	2.00	**1.000**	3.00	8-24 (0124568t)
4-8 (0156)	3.00	**1.000**	2.00	8-8 (01234789)
5-15 (01268)	1.00	**1.000**	4.00	7-15 (0124678)
5-19 (01367)	3.00	**1.000**	2.00	7-19 (0123679)
5-21 (01458)	3.00	**1.000**	2.00	7-21 (0124589)
5-28 (02368)	3.00	**1.000**	5.00	7-28 (0135679)
5-31 (01369)	1.00	**1.000**	4.00	7-31 (0134679)
5-33 (02468)	4.00	**1.000**	1.00	7-33 (012468t)
5-7 (01267)	2.00	**1.000**	3.00	7-7 (0123678)
6-Z17 (012478)	2.00	**1.000**	3.00	6-Z43 (012568)
6-Z44 (012569)	2.00	**1.000**	3.00	6-Z19 (013478)
6-Z49 (013479)	2.00	**1.000**	3.00	6-Z28 (013569)
4-23 (0257)	3.49	**0.897**	1.50	8-23 (0123578t)
6-33 (023579)	3.49	**0.897**		
3-9 (027)	1.01	**0.732**	4.01	9-9 (01235678t)
4-26 (0358)	4.01	**0.732**	4.01	8-26 (0134578t)
5-29 (01368)	1.99	**0.732**	3.00	7-29 (0124679)
5-34 (02469)	3.00	**0.732**	1.99	7-34 (013468t)
6-Z48 (012579)	1.01	**0.732**	4.01	6-Z26 (013578)
2-5 (05)	2.51	**0.518**	2.51	10-5 (012345789t)
3-11 (037)	2.51	**0.518**	2.51	9-11 (01235679t)
4-16 (0157)	2.51	**0.518**	2.51	8-16 (01235789)
4-20 (0158)	0.49	**0.518**	4.50	8-20 (01245789)
4-27 (0258)	2.51	**0.518**	2.51	8-27 (0124578t)
5-20 (01568)	5.51	**0.518**	4.50	7-20 (0125679)
5-30 (01468)	3.49	**0.518**	1.50	7-30 (0124689)
5-32 (01469)	1.50	**0.518**	3.49	7-32 (0134689)
6-18 (012578)	2.51	**0.518**		
6-31 (014579)	4.50	**0.518**		
6-34 (013579)	2.51	**0.518**		
6-Z50 (014679)	6.49	**0.518**	4.50	6-Z29 (023679)
5-35 (02479)	2.02	**0.268**	3.00	7-35 (013568t)
"0-1 ()"	0.00	**0.000**	0.00	12-1 (0123456789te)
2-6 (06)	0.00	**0.000**	0.00	10-6 (012346789t)
3-12 (048)	0.00	**0.000**	0.00	9-12 (01245689t)
4-9 (0167)	0.00	**0.000**	0.00	8-9 (01236789)
4-25 (0268)	0.00	**0.000**	0.00	8-25 (0124678t)
4-28 (0369)	0.00	**0.000**	0.00	8-28 (0134679t)
5-22 (01478)	0.00	**0.000**	0.00	8-22 (0123568t)
6-30 (013679)	0.00	**0.000**		
6-20 (014589)	0.00	**0.000**		
6-7 (012678)	0.00	**0.000**		
6-35 (02468T)	0.00	**0.000**		

augmented triads piled on. Curiously, no balance class has more than one prime of the same cardinality. It seems a given magnitude manifests in a given cardinality in only one pristine way. One balance class, magnitude = 0.00, has no primes. By definition, set classes with a magnitude of 0.00 in F(12,1) must be composed exclusively of 187ritons and/or augmented triads, with no remainder.

As a perusal of Figure 2.44 will reveal, some F1-complementation results in pc duplication. Figure 2.45 shows the F1-complementation of 5-10 (01346) and 4-4 (0125), but the set classes must be implemented as *mset classes* 01346 and 0125 in order to allow the doubling at pc 0. F1-complementation deals in pc-representatives, not pitch classes.

Figure 2.45 F1-Complementation of mset classes; union has pc duplication



● 4-4 (0125) <1,1,1,1>: F(12,1) = 2.394
○ 5-10 (01346) <1,1,1,1,1>: F(12,1) = 2.394

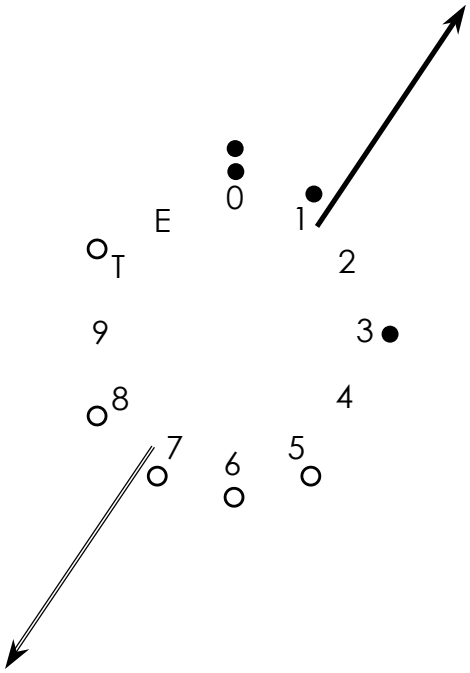
That is, each set class contributes a pc-representative of pitch-class 0. Two set classes would simply "share" pc 0, while two mset classes are allowed each to have its stake. This doubled weight at 0 is essential for maintaining balance. Furthermore, as can be seen in the figure, when demonstrating the union's balance by removing annihilating pairs and triples both pc-representatives are needed, one for a tritone and one for an augmented triad.

In much the same way that multiplicity affects a chord's interval-class vector, it also affects a chord's F(12,1) magnitude (and direction, for that matter) in interesting ways. For example, mset classes 0013 and 01235 have the same F(12,1) magnitude. (When doing the vector addition, 0013 would contribute two arrows in the direction of pc 0. On the metaphorical balance, there are simply two "stones" placed on the pan at 0.) So Quinn's theory, adapted to multisets, reads an affinity between the two. Their ICVs (1/212000 and 0/332110) prove their Z/Zo/Ze/Zoe-*un*relatedness, yet there is this affinity between the forces they exert in the chromatic F(12,1) space. Both mset classes also have an F(12,1) direction of xx.92, so F1-complementation is possible and is demonstrated in Figure 2.46.

One possible next step would be simply to list all multiset classes as well as the balance classes into which they fall. Preliminary work shows that some mset classes fall into the same balance classes as do the set classes, but there are many new balance classes (new F(12,1) magnitudes) as well. Continuing this exercise might produce some interesting results and highlight some unexpected affinities, but our task here is to find complementation, so it perhaps will save time to jump immediately to the msetclasses that are balanced in F(12,1) and treat them as the unions, or supersets, that two mset

classes may combine to perform. In other words, if the goal of identifying balances classes is to find F1-complementary pairs, it is more efficient first to find the available balanced supersets, the "aggregate proxies," and to let the analyst use them to find F1-complements of a given mset class.

Figure 2.46 F1-Complementation of mset classes; one mset class has pc duplication



● 0013: F(12,1) = 2.394
○ 01235: F(12,1) = 2.394

Figure 2.47 lists all mset classes that are balanced in F(12,1), i.e., that have the "exceptional property." The mset classes are sorted into columns by parent class and into rows by number of elements. (I chose arbitrarily to curb infinity at twelve elements.) Shown in boldface are those balanced mset classes that have the same number of

Figure 2.47 All balanced mset classes in F(12,1) up to twelve elements

	2-6	3-12	4-9	4-24	4-25	4-28	5-22	5-33	6-Z17	6-7	6-Z28	6-30	6-20	6-35
	06	048	0167	0248	0268	0369	01478	02468	012478	012678	013569	013679	014589	02468t
2	**06**													
3		**048**												
4	0066		**0167**		**0268**	**0369**								
5				02488			**01478**							
6	000666	004488	001667		002668	003669				**012678**		**013679**	**014589**	**02468t**
7				0224888			0114778	0024688	0124788		0135699			
8	00006666		00016667	00244888	00026668	00036669	00144788			00126678		00136679		0024668t
			00116677		00226688	00336699				01126778		01136779		
												01336799		
9		000444888		022248888			011147778	000246688	012247888		001356699		001445889	00244688t
									011247788		013356999			
10	0000066666		0000166667		0000266668	0000366669				0001266678		0001366679		00246668t
			0001166677		0002266688	0003366699				0011266778		0011366779		002246688t
										0012266788		0013366799		
										0111267778		0111367779		
												0113367799		
												0133367999		
11								00002466688	01112477778		00013566699			00002466688
								00022466888	01112477788		00133566999			0002446688t
									01122477888		01333569999			0022446888t
									01222478888					
12	000000666666	000044448888	000001666667	002224488888	000002666668	000003666669	001114477788	000024466888	001112477788	000012666678	001135566999	000013666679	000144458889	00002466668t
			000011666677		000022666688	000033666699			001124477888	000112666778	011335569999	000113666779	001144558899	00022466688t
			000111666777		000222666888	000333666999			001224478888	000122666788		000133666799		00024446888t
										001112667778		001113667779		0022446688tt
										001122667788		001133667799		
										011112677778		001333679999		
												011113677779		
												011133677799		
												011333677999		
												013333679999		
												013333679999		

	7-21	7-15	7-22	7-28	8-25	8-7	8-8	8-18	8-9	8-19	8-20	8-25	8-17	8-28
	0124589	0124678	0125689	0135679	0124678t	01234589	01234789	01235689	01236789	01245689	01245789	0124678t	01345689	0134679t
2														
3														
4														
5														
6														
7			**0125689**											
8	01245889				**0124678t**				**01236789**			**0124678t**		**0134679t**
9		001246788	001256689	011356799			012347889	012356899				01244678tt		
10	0122458889				00124678t	0123458899			0012366789	0012456889	0112457889	00124678t	0013456899	001346679t
					01124678t				0112367789			00124678tt		
					01244678tt							01224678tt		
												01244678tt		
11	00124458889	00012466788	00012566689	00113566799	00012467888		00123447889	00123566899				0012446788tt		
	01124558899	00012467888	00122566889	01113567799	0012446788t		01123477889	01223568899						
		00112467788		01133567999			01223478899	01233568999						
12	012224588889		001125566899	011135567999	00012466678t	012234588899		011235568999	000123666789	000124566889	011124577889	00012466678t	000134566899	00013466679t
					0012446678tt				001123667789	001224568889		00112466778t		00113466779t
					012444678ttt				001223667889			00122466788t		00133466799t
									001233667899			00124466788t		00134466779t
									011123677789			01112467778t		01113467779t
									011223677889			01124467788t		

	9-3	9-4	9-5	9-10	9-11	9-12	10-4	10-6	12-1
	012345689	012345789	012346789	01234679t	01235679t	01245689t	012345689t	012346789t	0123456789te
2									
3									
4									
5									
6									
7									
8									
9						**01245689t**			
10								012346789t	
					01123566799t				
11			00123467889			0012456689t	0123456899t		
				012346679tt					
12	001234568899	011234578899		00113466779t		001125566899		00123466789t	**0123456789te**
						00124456889t		01123467789t	
						01124556899t		01223467889t	

elements as objects; they "look like" regular set classes. This reveals some balanced mset classes that are off-sprung from non-balanced parent classes. For example, set class 4-24 (0248) is not balanced in F(12,1). It tips toward the two. (Annihilate the {2,8} and you're left with {0,4}.) Mset class 02488, however, an offspring of 4-24 (0248) is balanced. (Annihilate {0,4,8} and {2,8}.) Because all balanced mset classes in F(12,1) may be reduced to nothing by annihilating pairs and triples, these results were enumerated through a systematic combination as many T_n/T_nI forms of 2-6 (06) and/or 3-12 (048) as possible until all combinations up to twelve elements were exhausted. The list is somewhat unwieldy but far more manageable than the enormous list of all mset classes!

In practice, one might use the table in Figure 2.47 like this:

(1) What is one possible F1-complement of mset class 013?

(2) A member of 013 is found as a subset of the prime form 0**<u>124</u>**788

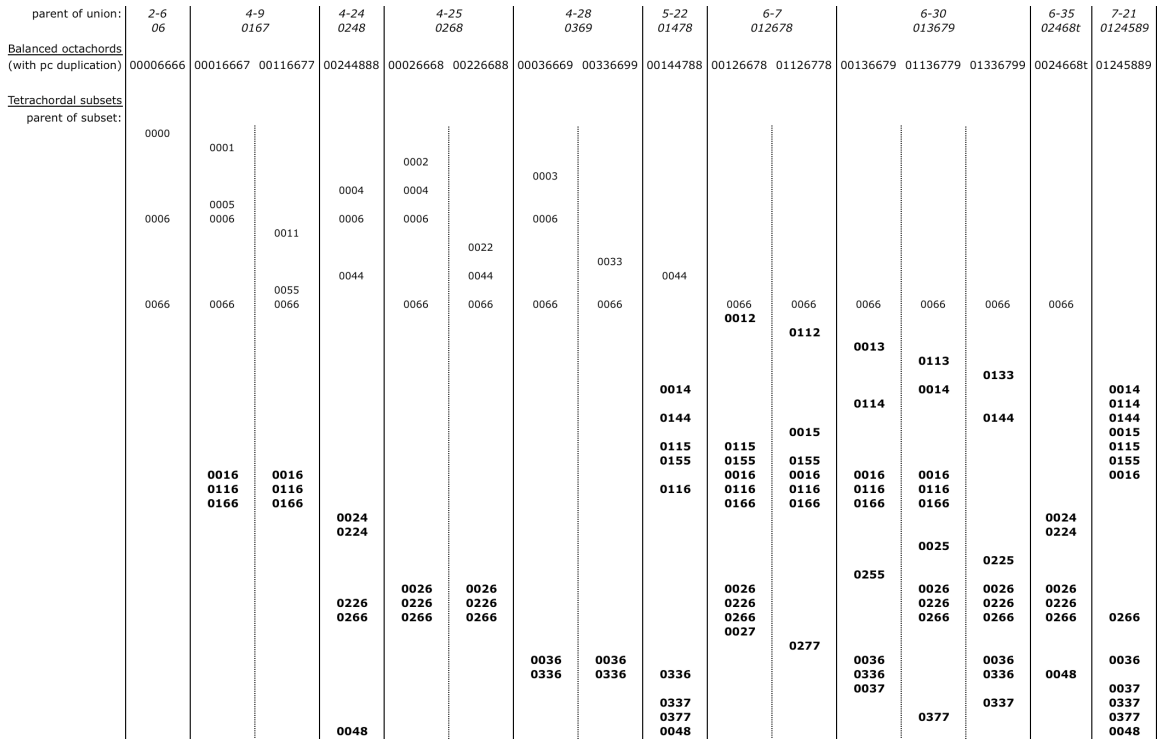
(see column "6-Z17 (012478)")

(3) The F1-complementary pcmset here is 0---788, a member of mset class 0115.

(4) Mset classes 013 and 0115 are F1-complements.

One must remember that an mset class may have several different F1-complements, each combining with the original mset class to form a different balanced superset. For this reason, one might prefer to limit an F1-complement search in some way, perhaps only to those potential partners that have the same number of elements, for example. The exercise to follow below takes this suggestion and examines, as a test case, only those balanced mset classes with exactly eight elements and looks for complementary pairs of 4-element mset classes.

Figure 2.48 All tetrachordal subsets (with pc duplication) of all F(12,1)-balanced octachords.



The top of Figure 2.48 is a cross section of Figure 2.47, a slice at the 8-element row that culls all of the balanced octachords, omitting those with no pc duplication and retaining the parent class headings at the top. The F(12,1)-balanced, 8-element mset classes are shown below their parent classes and each is trailed below by a list of all 4-element subsets of each.[61] Since the task in this particular exercise primarily is to find the 4-element F1-complements of the 3-object tetrachords (the 12 canonical trichords + 1 multiplicity: 0012, 0112, 0013, 0113, 0133, etc.), only the 3-object tetrachords are

[61] The 4-object tetrachords are omitted because, presumably, one would be interested in this information when finding the F1-complement of a mset class *with* pc duplication. With simple, traditional complementation one easily can find a 8-object, 8-element complement for any 4-object, 4-element tetrachord. Of course, this doesn't preclude the F1-complementing of a 3-object tetrachord with a 4-object one. It just means that, when using this chart, one would begin with the 3-object one.

highlighted in boldface. The 1- and 2-object tetrachords are included, however. Revealed here is that if a given mset class is to have an F1-complement with the same number of elements, there are a limited number of supersets it can unite with this partner to form. Of the thirty 3-element tetrachords, twenty-one can form with its F1-complement 1, 2, or 3 different balanced unions each. Some, more "promiscuous," can form 7 or 8 different balanced unions. Many of these 3-element tetrachords form F1-complementary pairs only with themselves. (See 0012 + 0012 to form 00126678, for example.) Others form more surprising pairs. (See 0266 + 0348 to form 01245889.) In any case, a table such as this one demonstrates that F1-complementation affords tidier though not always more exclusive, coupling than weighted complementation does. Once the analyst chooses a particular balanced mset as aggregate proxy, the field of possibilities is conveniently and significantly narrowed.

2.3.3 Negative Membership

Of the solutions to the problem of pcmultiset complementation, the last to be discussed here may be the most speculative, but it is the cleanest and most efficient. Like traditional complementation, and unlike most F1-complementation, it achieves the completion of a *single* (non-weighted) aggregate. Furthermore, unlike in weighted complementation, every mset class has exactly one complementary partner. No promiscuity. What is more, it works in precisely the same way as does traditional complementation, but to accommodate pc duplications it must use pcmsets (and mset classes) containing *negative multiplicity* of one or more of its members. That is to say at least one member of the pair must have a negative number of representatives of some pc.

Coming to grips with, say, -2 C#s in a pcmset is not easy, and what this negative membership means musically is an open question. One easily can comprehend a pcmset with no C#s, but a negative one? Some thoughts about how to conceive (if not to perceive) a negative pc are forthcoming, but first some examples should demonstrate, at the very least, the elegance of the concept in practice.[62]

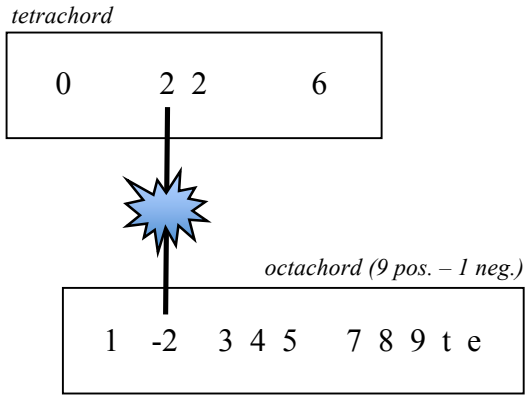
Following standard pcset practice, pcmset {0,2,6} would have the following complementary pcmset: {1,3,4,5,7,8,9,t,e}. Together, the two pcsets complete a single aggregate with no "extra pcs" or doubling. If the first pcmset instead had two representatives of pc2—{0,2,2,6}—the remaining tones, {1,3,4,5,7,8,9,t,e}, would complete the aggregate, but the union would have 13 pc representatives, an "extra" pc2. To maintain the completion of a single aggregate there would need to be a way in which that extra pc is extinguished after the union of the pcmsets. The complement of {0,2,2,6} would need a "negative" pc2, an **anti-pc**. When united in a pcmset with pc2, this anti-pc2 would effectively cancel out pc2; the two would annihilate one another. Earlier, we considered annihilation of two tones that are balanced in F(12,1), in which the balanced tritone pc3 and pc9 remain in the pcset after their annihilation. (The pcs remain; it is only their forces that are annihilated.) Here, however, the annihilation would fully removes pc and anti-pc from existence! The anti-pc is a full-fledged member of pcmsets, but registers a negative multiplicity or "charge" for its particular pitch class. Take, for example, a pcmset with two pc3s and negative one pc5 (the anti-pc). In power notation, the pcmset would be notated thus: $\{3^2,5^{-1}\}$. In long form, like so: {3,3,**5**}, with its anti-pc highlighted. If another pc3 were added to the mset, the notation simply would read

[62] Negative membership in pcmsets was directly inspired by Wayne Blizard, "Negative Membership" *Notre Dame Journal of Formal Logic* 31/3 (Summer, 1990): 346-368. Blizard's allusions to the positron and dark matter in particle physics are particularly intriguing.

{3,3,3,***5***}; the multiplicity of that pc increases from 2 to 3. If a pc5 were added instead, the result would read {3,3}; the multiplicity of pc5 increased from -1 to 0. [63]

Our original pair of pcmsets appears, now with anti-pc2, in Figure 2.49. In a union of the two pcmsets, one of the first pcmset's pc2s and the other pcmset's anti-pc2 annihilate each other, leaving then necessary lone pc2 for the aggregate. (2 + (–1) = 1). The cardinality of each set is also indicated in the figure. Note that {1,***-2***,3,4,5,7,8,9,t,e} is an octachord. It contains nine pcs and one anti-pc summing to 8 elements altogether. There are ten objects in the pcmset, to be sure, but one object (2) is imbued with a negative charge, and nine positive. Objects (pcs) are neither positive nor negative; only the elements, which point to these objects, have such a charge. When we count the

Figure 2.49 Negative and positive pcs annihilating each other, leaving aggregate



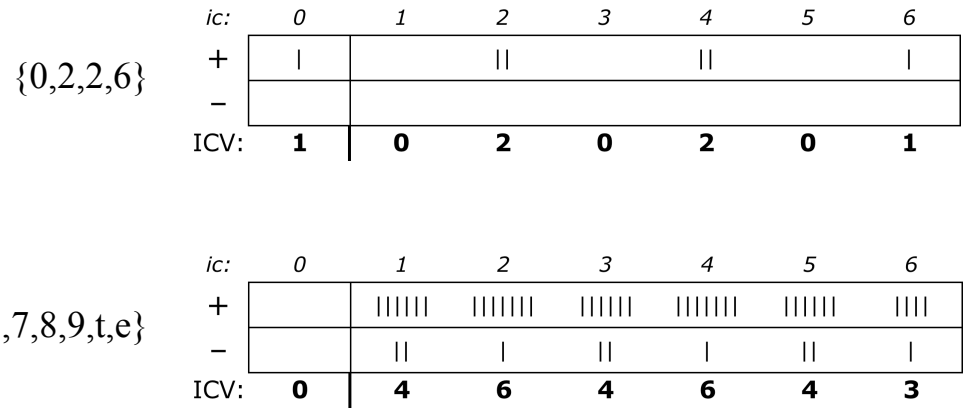
elements in a pcmset, we find its net element-charge, doing the simple arithmetic.

Before attempting interval-class vectors, we first must decide how to refer to an interval between a pc and an anti-pc. If anti-pcs exist in some dark world, alongside us

[63] An efficient way to represent pcmultisets with positive, null, or negative multiplicity is to use a 12-position vector, where each position displays multiplicity of a different pc, 0-11. Here is how the pcmsets in Figure 2.49 would be represented: 102000100000 and 01-1111011111. Summing the two vectors, of course, results in 111111111111, the aggregate. These are, essentially, mfunctions operating all 12 pcs.

yet invisible (inaudible), the measured distance between them and standard pcs ought to reflect the usual interval classes but be designated appropriately. I propose that the interval class formed by pc x and anti-pc y is measured in the usual way (y – x (mod 12)), but that its value is negative; it is a negative interval class. If the ic formed by pc2 and pc7 is ic5, then the ic formed by pc2 and *anti*-pc7 is ic –5. It follows, then, that the ic between two anti-pcs is always positive. This means than an ic-tally in the calculation of interval-class vectors must account for both positive and negative interval classes before summing.[64] Figure 2.50 again displays our complementary pair, but also includes an interval-class tally for each pcmset. The additional row gives us room to tally ics both positive and negative, and the sum of all intervals is in the bottom row comprising the ICV. A pcmset may contain some negative and positive instances of any ic except ic0. Because ic0 is formed between two representatives of the same pc, a negative ic0 is impossible within one pcmset. For example pc4 and anti-pc4 might seem to form a

Figure 2.50 Interval-class vectors with negative and positive tallies

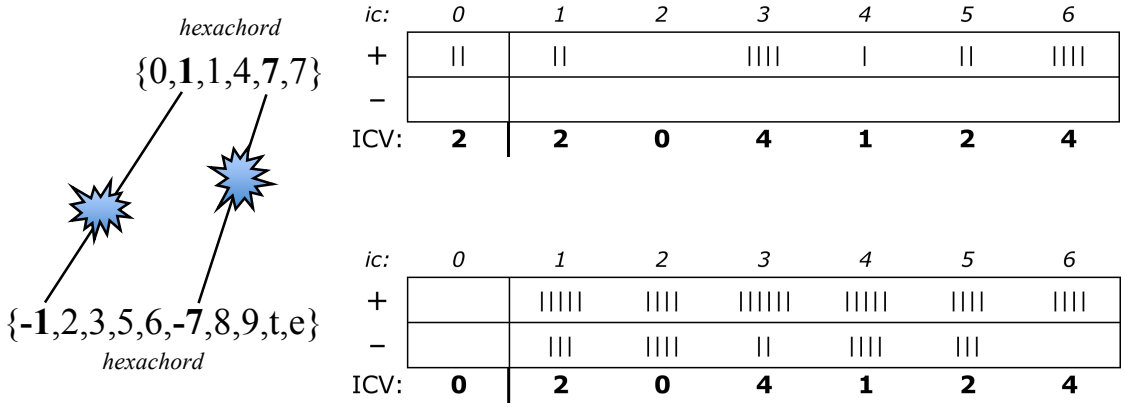


[64] This is not unlike the multiplication of positive and negative numbers: positive x positive = positive; positive x negative = negative; negative x negative = positive. The negative results only when combing two items of opposite polarity.

negative unison, but if the two were in the same pcmset, they would annihilate each other out of existence before their unison could be measured.

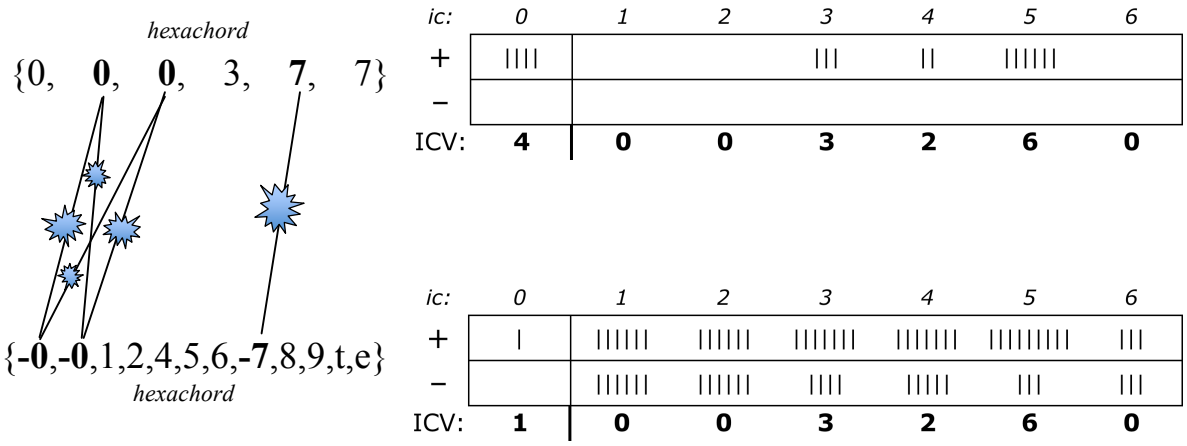
A comparison of the two ICVs in Figure 2.50 reveals that the two are in the same relationship as traditional tetrachord/octachord complements. That is to say one can add the vector 444442 (or, here, 0/444442) to the ICV of the tetrachord to produce the ICV of the octachord. This remarkably holds true, no matter how many anti-pc are accounted. Further, it seems to confirm that {1,**-2**,3,4,5,7,8,9,t,e} is behaving like an octachord (with ten elements, nine positive and one negative. How to account, then, for the lone ic0? Continuing my speculation, I suggest that the total, which always will be positive, of ic0s in the two sets is a kind of "residue" of annihilation. It accounts for the exact number of annihilating pairs in the union. Figure 2.51 offers another demonstration. It contains two complementary hexachords, whose ICVs as one might expect are identical. The residue in the ic0 column once again accounts for the two annihilations.

Figure 2.51 1-Aggregate complementation of pcmultiset with pc duplication



This procedure works just as well for multiplicities greater than two. Figure 2.52 shows pcmset {0,0,0,3,7,7} and its complement. These are two hexachords again, but this time each contains at least one unison. The five total unisons in the ICVs represent the residue of five potential annihilations.[65]

Figure 2.52 1-Aggregate complementation of pcmultiset with pc triplication

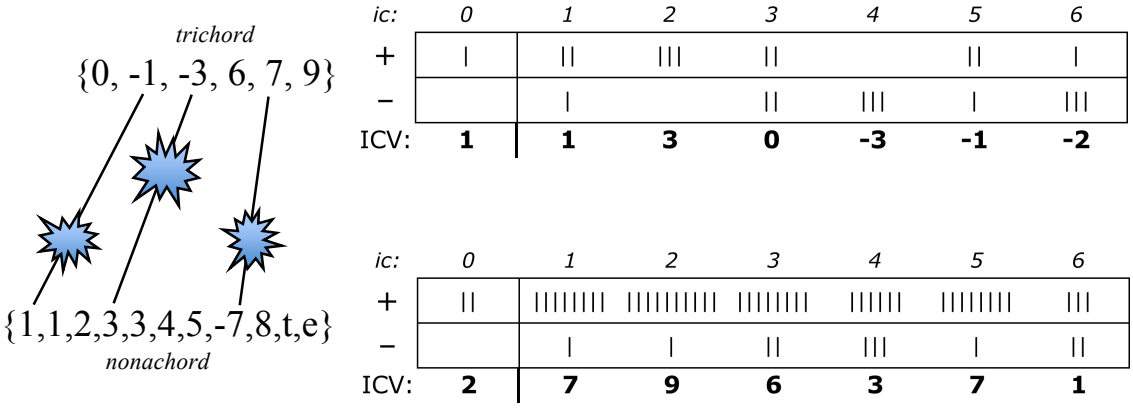


In each of the previous cases, the first set contained only pcs, no anti-pcs, leaving the complement to provide them. We can, however, find complements of pcmsest of any configuration, even those with pc duplications *and* anti-pcs. In Figure 2.53, the pcmset {0,-1,-3,-6,7,7,9}, which is a triad, and its complement {1,1,2,3,3,4,5,-7,8,t,e}, a nonachord, are shown with their ICVs. Note that even though the first has negative values in its vector, the difference between the two vectors is 0/666663, just as expected between triads and nonachords. In this case the presence of multiple anti-pcs results in a

[65] I say potential annihilations because it is counterintuitive to suggest that one entity annihilates with two others in the same magnitude as it would with one with one. What should be apparent by now, though, is that "annihilations" actually are the "negative unisons" prohibited in a single pcmset, and between these two particular pcmsets there are five such intervals.

negative total of intervals in one of the pcmsets. If we accept the anti-pc, we must similary accept the "anti-interval" as well as any ICV that registers a negative number of some interval class.

Figure 2.53 1-Aggregate complementation, each pcmultiset having negative membership



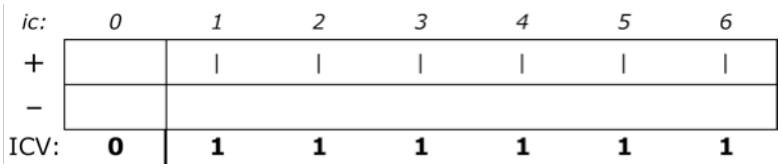
Complementation in the preceding examples always summed to exactly one aggregate. This was called 1-aggregate complementation. With this method, though, one can easily change the intended number of formed aggregates to two, three, or more. Summing to two aggregates in this way, as described at the beginning of Section 2.3, would require no anti-pcs at all if the pcmsets themselves have multiplicity no higher than two. A final, interesting possibility is the complementing of pcmsets to form exactly 0 aggregates. In 0-aggregate complementation, a pcmset gets coupled with its exact opposite, a pcmset with the same pc integers but with reversed polarity.

Figure 2.54a shows pcmset {2,3,6,8} and its ICV. The 1-aggregate complement, of course, is {4,5,7,9,t,e,0,1}. Its 0-aggregate complement, however, would not need pcs 4, 5, 7, 9, etc. because the first set already contributes none of these, and our goal is to

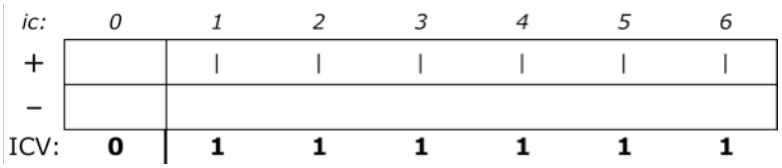
Figure 2.54 0-Aggregate complementation, or "shadow pcmsets"

(a)

{2, 3, 6, 8}

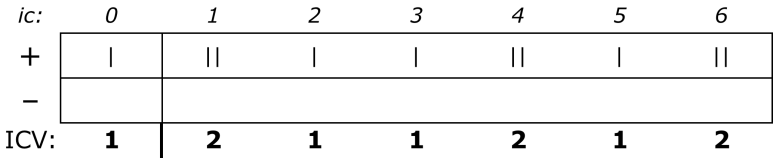


{-2, -3, -6, -8}

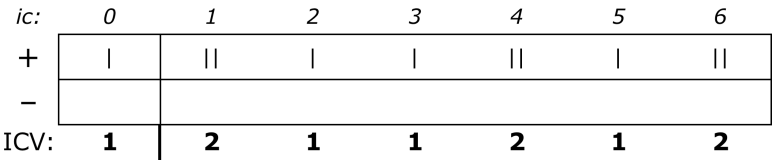


(b)

{2, 2, 3, 6, 8}

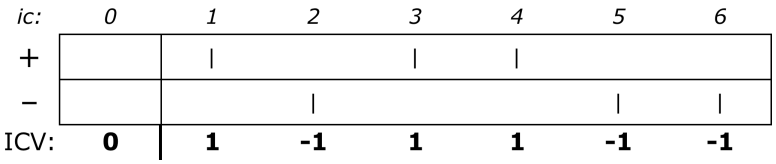


{-2, -2, -3, -6, -8}

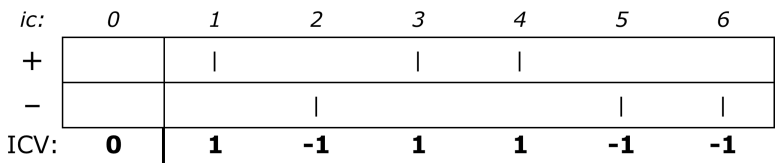


©

{2, 3, 6, -8}



{-2, -3, -6, 8}



have none of each pc in the union. No, the complement needs to provide an anti-pc for each of the pcs in the first pcmset. The pcmset {-4,-5,-7,-9} would do the trick. As can be seen in the figure, each pc is annihilated, and their ICVs are the same.[66] In 0-aggregate complementation the cardinality of pcmsets in the pair sum not to 12, as in 1-aggregate complementation, but to 0. The pcmsets in Figure 2.54a are a tetrachord and an anti-tetrachord, containing 4 pcs and -4 pcs respectively. The pcmset {-0,-1,-4,-6} acts as a **shadow pcmset** to {0,1,4,6}, as if exsiting right alongside its partner but remaining invisible.[67]

As a variation on this theme, Figure 2.54b shows pcmset and its shadow pcmset, {-0,-0,-1,-4,-6}. If a pcmset contains both positive and negative pcmembership, then its shadow will, too. Figure 2.54c shows pcmset {0,1,4,-6} and its shadow.

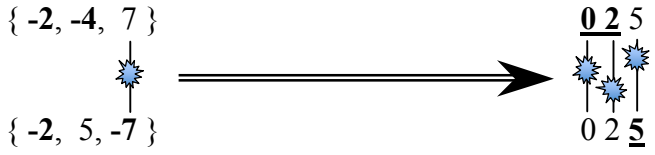
One might, in critique, fairly suggest that in practice any two pcmsets can be shown to be 0-aggregate complements, or shadows, of one another so long as the appropriate negative pcs are appended. For example, {0,2,3} and {1,4} can be seen as shadows if we assume that they "really" are {0,-1,2,3,-4} and {-0,1,-2,-3,4}. If these anti-pcs cannot be heard anyway, why not assume that they are there, somehow bringing these two pcmsets into balance? This would not be the case, however, for, say X = {0,1,2} and Y = {2,4}. One could not simply add an anti-pc 2 to X in order to annihilate the pc2 in Y, because it would annihilate the neighboring pc 2 in X. In pcmset a pc can have only one multiplicity: negative, 0, or positive; there cannot be -1 *and* 1 of a pc. The

[66] In 0-aggregate complementation the ic0 does not indicate annihilation residue but simply the number of unisons in each pcmset.
[67] The term "shadow pcmset" is a borrowing of Blizard's term, "shadow mset." See "Negative Membership," p. 349.

criticism, though, will be worth considering once—if ever—one successfully "finds" and details the anti-pc itself.

An extension of this idea, not to be explored in depth here, is the abstract complementation of mset classes instead of the literal complementation of pcmsets. Returning to the pcmsets in Figure 2.51, one finds that {0,1,1,4,7,7} and {-1,2,3,4,6,-7,8,9,t,e} belong to mset classes 011477 and 0123**4**5789**t**, respectively.[68] (Due to mset-class notation's not benefitting from the spacing that commas provide, the potentially confusing minus signs are replaced by bold face and underlines.) The annihilating pairs are not as immediately apparent here as they are in pcmsets and literal complementation, but with the appropriate transposition or inversion, one can spot them. The annihilation does not occur between actual pcs, but at one level of abstraction. The same is true for 0-aggregate completion. In Figure 2.55, pcmsets X = {-2,-4,7} and Y = {-2,5,7} clearly are not shadow pcmsets; their union is not the 0-aggregate. Altogether there are -2 pc2s, -1 pc4, 1 pc5, and 2 pc7s. When viewed as mset classes, however, the two are indeed shadows: **02**5 and 02**5**. Looking back at the pcmsets in the figure, we see that the anti-pc2 in X annihilates the pc7 in Y, but the annihilation is taking place abstractly. Pcmsets

Figure 2.55 Shadow multiset classes



These ***are not*** shadow pcmsets… …but ***are*** shadow mset classes

[68] Finding normal form is done in the same way, and polarity of each pc is retained through the necessary transposition and inversion.

025 and 02**5** complement one another regardless of their specific manifestation in pitch space. This is simply the application of the principle of abstract complementation to 0-aggregate complementation.

Notwithstanding the tidiness of the foregoing, a larger question still looms: exactly what is an anti-pc? How can a pcmset contain "negative two" F#s? One interpretation of an anti-pc would explain it as a note promised in some way—by other tones, cadential formulae, or some other precedent—but ultimately withheld. The cadential F natural that Edward T. Cone's "promissory" E natural anticipates, might—at least, that is, if it never were to appear at all—serve as an example.[69] Such cases, though, imply that something is altogether missing and that, furthermore, it something separate from that which demands it. An F natural might falsely be promised by an E natural, but one is unlikely to consider the two notes together in the same collection. It is because one demands the other that one would tend to view them as separate entities and not two disinterested members of a single collection. Whatever effect, or meaning, produced by an expected, but missing, tone must be different from that produced by a true "anti-tone"; zero multiplicity must somehow be stronger (negatively) than 0 multiplicity. I envision the anti-pc as an active, independent (anti-)tone, one whose reason for being comes not from another pc, cadence, pc sets, or other context but from its own negativity. Standing alone, the pcmset {2,-4,7} should in someway be different from {2,7} regardless of what the surrounding material is promising, requesting, or expecting. How such a concept will be realized remains unseen.

---

[69] Edward T. Cone, "Schubert's Promissory Note: An Exercise in Musical Hermenuetics" *19th-Century Music* 5/3 (Spring, 1982): 233-241. I'm grateful for a brief conversation with Richard Cohn and David Clampitt who, upon my raising the notion of a "negative" pc, referred immediately to this article.

The anti-pc, if it exists, may not be directly measureable at all. Perhaps it will be detected only by reading effects it has upon nearby pcs, evidence of its mysterious pull. Another possible avenue, however, is to adopt positive and negative charges as metaphor. It is difficult to imagine a note "not existing" much less a note existing in negative number. Much easier would be to allow the listener to experience the metaphorical interaction of both positive and negative charges. The analyst would assign two different timbres or instrumental groups to the "positive" and "negative" realms. Then, the music may be seen to enter in and out of 0-complementation as the two timbres cancel each other out. The anti-pc, then, would not remain dark, mysterious, unexplained, and unseen. Rather, the audience would be able to watch omnisciently and hear the cosmic drama unfold.

ANALYTICAL INTERLUDE #2

Fourier (Im)balance in Bach's *Chorale #83*

Figure 2.56 contains the first two measures of J.S Bach's Chorale #83, "Jesu Leiden, Pein und Tod."[70] There are seven consecutive chords, numbered here accordingly. Figure 2.57 plots those seven chords as pcsets in a two-dimensional **balance space** whose axes represent the force and magnitude of F(12,1). Since chords 1, 3, 4, 5, and 6 all are equally balanced major or minor triads (i.e. set-class 037) they lie at the same magnitude: 0.518 lewins. They have different directions, of course, because they are *different* major and minor triads. As one might expect, a passage of triadic harmony shifts little in magnitude, but quite a lot in direction. In other words, the qualities are the same, but the roots shift. Notice also, that even the dominant seventh chords have the same magnitude as the major and minor triads. The progression's path is traced from chord to chord with arrows.

If we interpret the chords as multisets, the balance space springs dramatically to life. Figure 2.58 models the same excerpt as does Figure 2.57 but tracks the chords as pcmultisets instead of pcsets. Suddenly a major triad with a doubled fifth is quite different in magnitude, or quality, from one with a doubled root. (Compare chords 4 and 5.) On the other hand, a minor triad with a doubled root (chord three) registers the same magnitude as a major triad with doubled fifth (chord four). They are, after all, inversions of one another. Figure 2.58, in general, involves higher magnitudes because all chords

[70] This example and analysis first appeared in Thomas Robinson, "Pitch-Class-Balance Space" Graduate Students in Music Symposium, The Graduate Center, CUNY, New York, NY (April 2006).

Figure 2.56 J.S. Bach, Chorale #83, *Jesu Leiden, Pein und Tod*

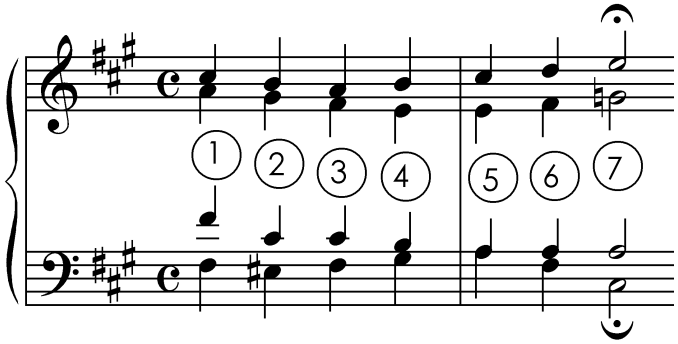


Figure 2.57 Bach's harmonic progression expressed as F(12,1) qualities. (Chords interpreted as pcsets.)

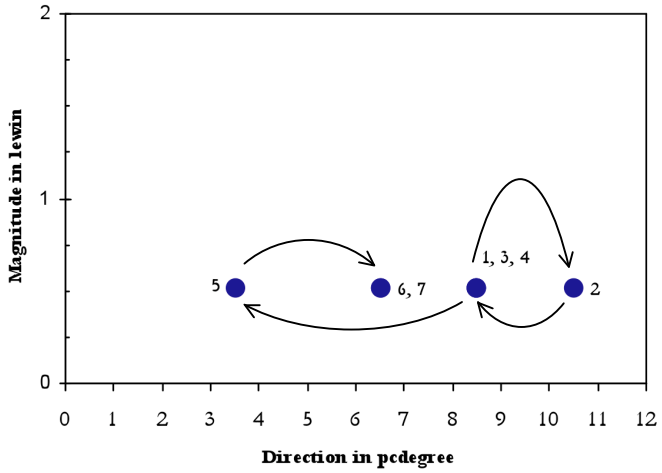
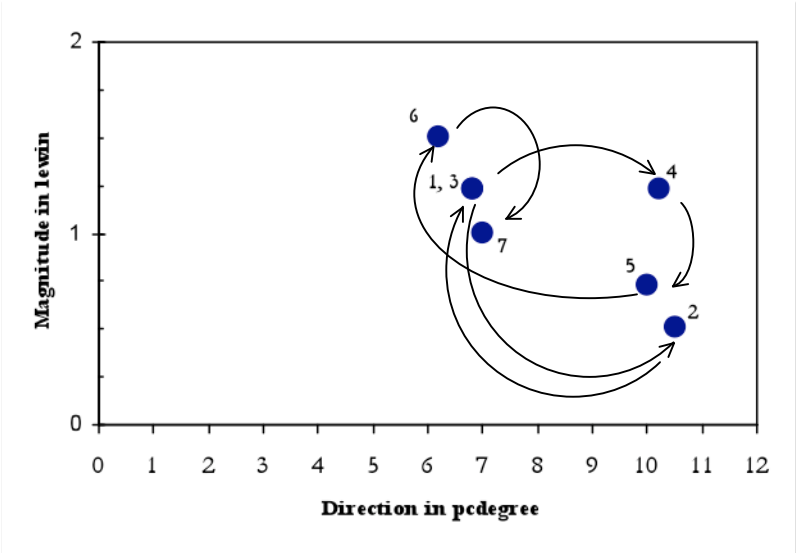


Figure 2.58 Bach's harmonic progression expressed as F(12,1) qualities. (Chords interpreted as pcmultisets.)



are now technically tetrachords—all four of Bach's voices are accounted for. There are more pitch classes in play, increasing the overall potential force. The only chord whose magnitude is as low as it was in the previous figure is chord 2, a dominant seventh, which was a tetrachord in the first place. My assertion here is that pcmultisets offer a more dynamic picture of the changing qualities of chords than do traditional pcsets. In other words, not all major triads were created equal. It might be worthwhile to consider multiple inflections of or variations on a single triad.

2.4 Conclusion

Sometimes at great length, the first two chapters that form the body of this dissertation introduced pcmultisets; considered how to label and count them; offered ways to assess and compare them; and even questioned whether or how they exist. After each chapter I made a modest step toward a possible avenue of analysis, toward ameaningful application of our newly developed multiset conceptualization.

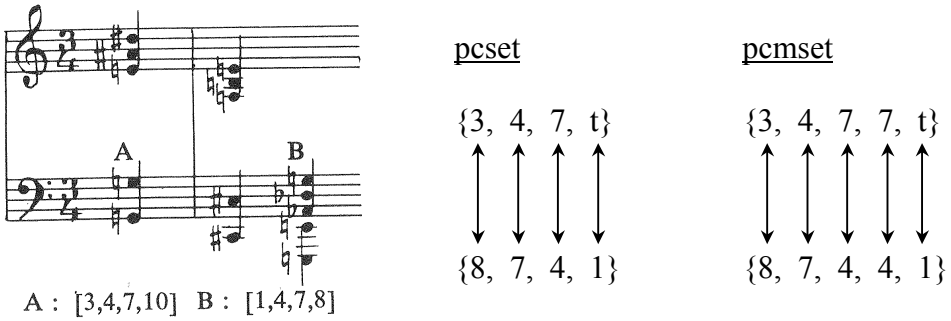
The brief analytical interludes between chapters suggested the existence of relationships and transformations previously unnoticed or unheard, as if lying beneath the surface undetected until the arrival of multisets. And this is a profitable approach one might take with pcmultisets. Specifically, one might explore well-known and oft-studied pieces keeping an eye and an ear open for pitch- and pitch-class doubling. One might find new ways to hear and to understand the classic repertoire. Even classic analyses that have become graduate school staples could benefit from pcmultiset analysis; their claims, in turn, would possibly be confirmed, modified, or even rejected.

Most fundamentally, however, the pcmultiset stands aside the pcset, and we may simply profit from comparing the two. When we apprehend some instance of multiple pitch tokens of the same pitch type, from this information we may infer a pcmultiset and even a multiset class. We might find that the particular mfunction exhibited by the mset class is retained through the chord's transformation. Perhaps the mfunction itself is transformed. Whatever the case, we would hope that accounting for doubling tells us something of interest that ignoring doubling does not. *And that is the basic goal of pcmultisets.* The effort to add the extra elements to our basic set, to construct a new

vector, and to manage an infinite number of mset classes is a substantial investment and demands a comparable payoff. While I hope this dissertation will lead to such rewards, there is only room here for a modest one.

The two chords in Figure 2.59, from Schoenberg's Opus 15, No. 6, appear very early in Allen Forte's *The Structure of Atonal Music*, where they stand as an example of inversional equivalence.[71] The doubled pitch classes in the left-hand part are, of course, ignored and the pitch-class sets are shown below: {3,4,7,t} and {1,4,7,8}. With an index of 11, the inversional mappings are clear and are highlighted in the figure. Pcs 3 and 8 map to one another, 4 and 7, and so forth. If the two collections get treated as pcmultisets and the

Figure 2.59 Allen Forte's pcset inversion treated as pcmset inversion. From *The Structure of Atonal Music*.



doubled pcs are included, these transformation mappings are reinforced, if not "confirmed" in some way, by the mapping of the doubled pc7 and the doubled pc4 under this inversion. Had this been transposition, such a mapping of a doubled pc might seem routine, but in pitch-class inversion, which, itself, for some critics is a transformation of dubious "perceptibility," the multiplicity acts as a marker, emphasizing here a single

[71] Forte, *Structure*, p. 9.

member of the mset class and orienting the listener who may be disoriented by inversion. Because of consistent doubling, both pcmsets are members of the mset class 01447. If Forte's pcsets help to assert equivalence, then pcmsets make that assertion stronger.

Surely both the analytical and the theoretical literature is filled with similar cases or, on the other hand, cases where multiplicity and its mappings are more likely to complicate than to clarify. Whether or not to "use pcmultisets" often will depend on what in a given excerpt is considered the "line" and what is the "voice." If the guitar doubles the harp, note for note, should they be treated as pc doublings, or do the two instruments act, united, as a single voice? What about a whole violin section? Are they articulating dozens, if not hundreds, of unisons in a giant multiset?[72] These questions, of course, will be answered on a case-by-case basis, and the answers will vary, but they are not essentially different from many other routine analytical questions: how do we approach segmentation, what constitutes voice and voice leading, and so on. To add doubling to the puzzle is simply to add another dimension.

It ought to go without saying that pitch-class multisets and multiset classes are not improvements upon nor are they replacements for traditional set-theoretical tools. They simply stand at the ready when needed, either to emphasize a particular doubled tone or to study multiplicity. I would expect research in multisets to continue productively in the fields of voice-leading spaces, mathematical modeling, and even in the classic literature, whether tonal, post-tonal, or even pre-tonal. I humbly presume that this dissertation asks more questions than it answers, and that it stimulates interest in all who read it, encouraging further study and exploration.

---

[72] I thank Stephen Peles for his issuing a warning of this nightmare scenario.

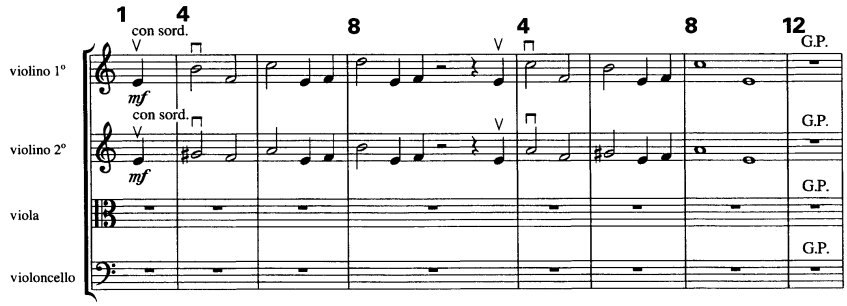
I leave the reader with one final analysis, in which I consider that transformations don't have to take place at the pitch or pitch-class level at all, and that the very mfunction itself is fertile ground for diverse transformational growth. Arvo Pärt's *Psalom* is composed of a single, reiterated pcset, but the multiplicities of its members rise and fall, the mfunctions rarely constituted the same way twice. It is perhaps in this arena, minimalism and "repetition," that pcmultisets might find a surprising, yet fitting, home.

ANALYTICAL POSTLUDE

Transformational Multiplicity in Pärt's *Psalom*

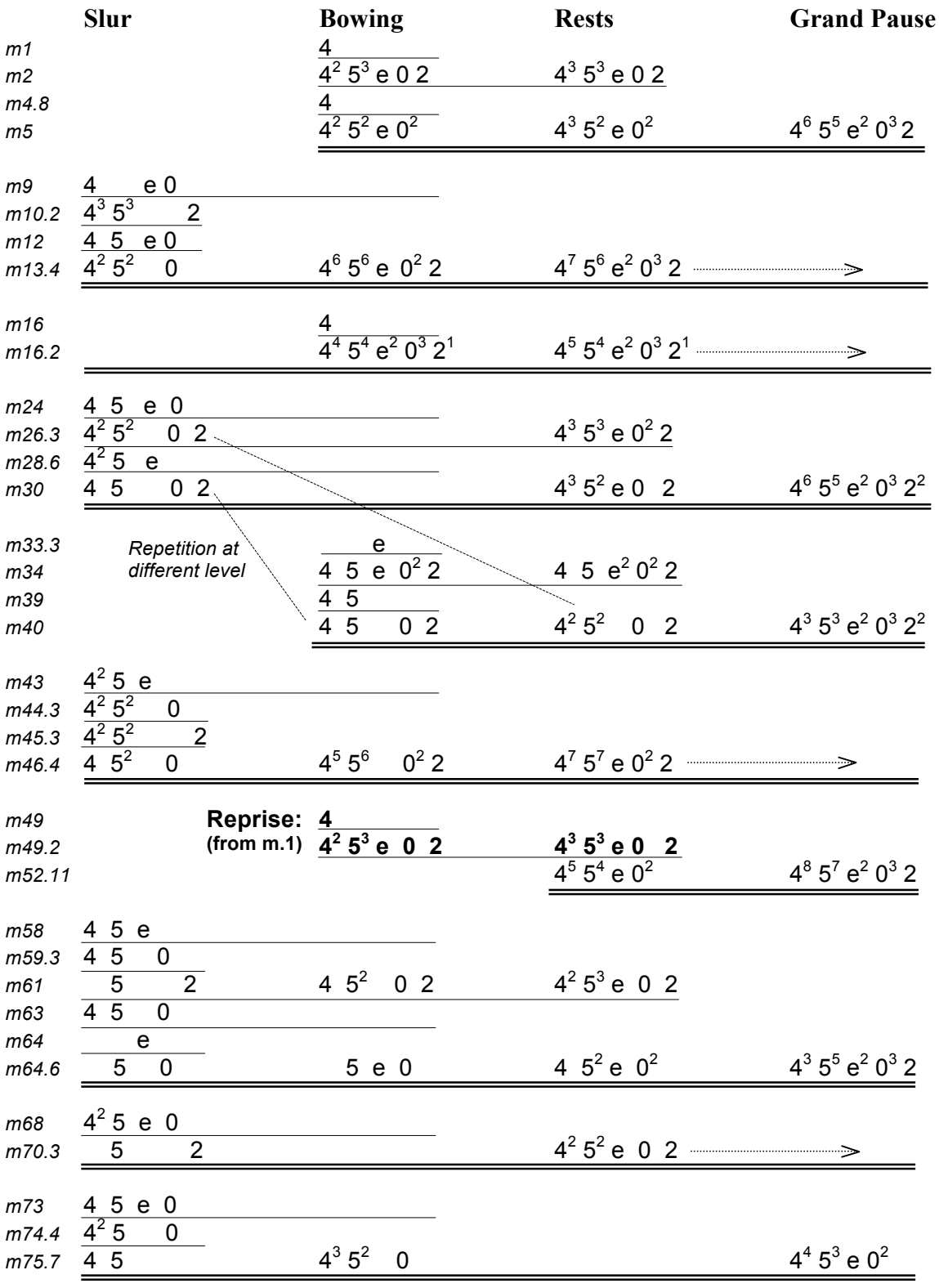
Arvo Pärt's *Psalom* for string quartet is a short piece maintaining a consistent texture. The opening passage, excerpted in Figure 2.60, is a fair microcosm of the piece as a whole. Apart from the pair of heterophonous voices, found in various instrumental pairs, there is only an intermittent drone supplied by one or both of the remaining instrument parts. In this small excerpt one also can see some of the various markers by which the piece is systematically segmented: the bowing, the rest, and the grand pause.

Figure 2.60 Arvo Pärt, *Psalom*, mm. 1-7



All melodic activity occurs only in this homophonic pair of voices. Further inspection will show that the lower voice of the pair always plays E, G#, F, A, and B whenever the upper voice plays E, B, F, C, and D respectively. An analysis of relative multiplicity of pc in one voice, then, will be identical to that in the other. Therefore, I attend only to the upper voice of the pair. Figure 2.61 shows melodic activity traced through the entire piece and partitioned hierarchically either by articulation or rest. The smallest, or most local, formal level includes notes grouped by a slur. A larger grouping,

Figure 2.61 Arvo Pärt, *Psalom*. Off-sprung pcmultisets of parent pcset {4,5,e,0,2}. Segmented hierarchically by slurs, bowings, rests, and grand pauses.



which may included more than one group partitioned by slur, is found under a single bowing. Larger still is the collection of notes found between rests. Finally the highest-level formal marker is that of the grand pause. Thus, the multiplicities found in the far-right column represent a sum of all multiplicities, the smaller multisets, within the boundaries marked by grand pauses. The left hand side is the most local level. As we move to the right, we get a more background, comprehensive view. The same pitch classes—4, 5, 11, 0, and 2—are present throughout, but their multiplicities, shown in power notation, are different in nearly every section, at nearly every level. Only a handful of specific multiplicities are repeated in the piece, four of which are labeled. See measure 30 in the example; the multiset found at the slur level is reproduced at the bowing level in measure 40.

Figure 2.62 is simply a restatement of Figure 2.61 but with power notation replaced by a concise pair of numbers. The first in the pair represents number of objects, the second the total multiplicity, or number of elements. For example, the multiset in measure 2 at the level of rest would be read as "a five-object nonachord." This example allows one to see the varied multiplicity in a quicker glance by comparing the second, boldfaced, numbers.

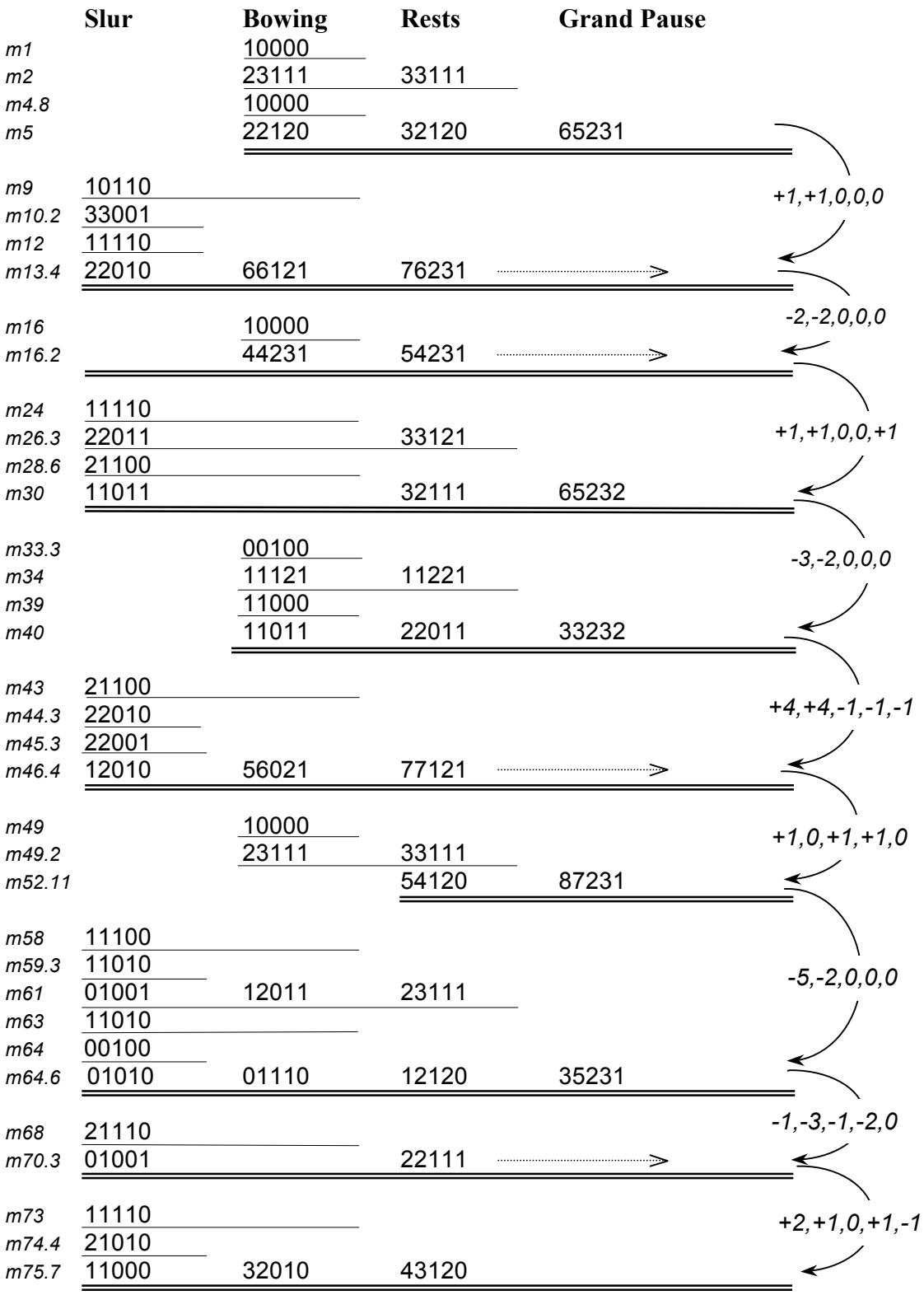
In Figure 2.63 each multiset appears again but is written as an mfunction.[73] This mfunction accounts only for multiplicities (the superscripts), and not the pcs themselves. It is always applied to the parent pcset {4,5,11,0,2}, the master set, so to speak, of the piece. In an analysis like this, where pc content remains the same, but multiplicity changes, this notation lets us focus only on the multiplicity and not the pcs themselves. A

[73] Angle brackets and commas are removed for clarity.

Figure 2.62 Arvo Pärt, *Psalom*. Pcmultisets are listed by # of objects / cardinality. (E.g., 3/**5** is a three-object pentachord.)

	Slur	Bowing	Rests	Grand Pause
m1		1/**1**		
m2		5/**8**	5/**9**	
m4.8		1/**1**		
m5		4/**7**	4/**8**	5/**17**
m9	3/**3**			
m10.2	3/**7**			
m12	4/**4**			
m13.4	3/**5**	5/**16**	5/**19**	→
m16		1/**1**		
m16.2		5/**14**	5/**15**	→
m24	4/**4**			
m26.3	4/**6**		5/**10**	
m28.6	3/**4**			
m30	4/**4**		5/**8**	5/**18**
m33.3		1/**1**		
m34		5/**6**	5/**7**	
m39		2/**2**		
m40		4/**4**	4/**6**	5/**13**
m43	3/**4**			
m44.3	3/**5**			
m45.3	3/**5**			
m46.4	3/**4**	4/**14**	5/**18**	→
m49		1/**1**		
m49.2		5/**8**	5/**9**	
m52.11			4/**12**	5/**21**
m58	3/**3**			
m59.3	3/**3**			
m61	2/**2**	4/**5**	5/**8**	
m63	3/**3**			
m64	1/**1**			
m64.6	2/**2**	3/**3**	4/**6**	5/**14**
m68	4/**5**			
m70.3	2/**2**		5/**7**	→
m73	4/**4**			
m74.4	3/**4**			
m75.7	2/**2**	3/**6**		4/**10**

Figure 2.63 Arvo Pärt, *Psalom*. Pcmultisets are listed by multiplicity vector. Multiplicity difference vectors are applied to pcmsets at the grand-pause level.



multiplicity difference vector can quantify the changes in multiplicity, and thereby represent a transformation of multiplicity. Down the right-hand side of Figure 2.63, five-position vectors trace the changes in multiplicity at the level of grand pause from pcmultiset to pcmultiset. The entry in each vector place represents change (+ or -) in the multiplicity of that particular object. Take, for example, the last two multisets at the grand-pause level in the lower right. The penultimate multiset is represented with multiplicity vector <2,2,1,1,1>. When subjected to the multiplicity difference vector <+2, +1, 0, +1, -1>, each entry simply adds to (or subtracts from) its respective multiplicity. This produces the final multiset's multiplicity function: <4,3,1,2,0>.

What is impressive about the data in these last three figures is the rich variety of multiplicities a single parent pcset can exhibit in a single piece. Far from a monotonous rumble, the piece shimmers in a non-repetitive manner throughout. One needs only know where to listen, for the interest lies not in the pcs themselves, as much as it does in their number. This, perhaps, is the strength of the pitch-class multiset in the end. If we aim to dig into the set itself, exploring its multiplicities and shading, we take the pitch classes for granted to a certain degree. Music like Pärt's often does exactly that; a collection of tones is presented as a given template or backdrop, and the activity bustles within.

APPENDIX 1

Due to size constraints, this 574-page appendix is stored online. As of July 2009, it can be found here:

https://wfs.gc.cuny.edu/TRobinson/www/dissertation.htm

Introductory remarks, however, are given below.

MULTISET CLASSES AND THEIR IC VECTORS

Pitch-class multisets may contain an unequal number of objects and elements. The number of elements is the set's cardinality, and the number of objects is the number of different pcs contained.

The following table lists all multiset classes up to cardinality twelve. (There is an imposed maximum of twelve elements.) The data is first grouped according to number of objects (pcs), then ranked according to number of elements. At the far left of each page are the multiplicity vectors. Each position in the vector represents a representative pc in the *parent class* (traditional set class) listed atop each column, and each value represents multiplicity of that pc. For example, 3-1(012) is the parent class of multiset 000122, which can be represented as $0_3 1_1 2_2$. The subscript, representing multiplicity of each object, can be extracted for use as the independent mfunction <3,1,2>. This vector then can be applied to any three-object parent, i.e. to any of the traditional trichord set classes.

Under each parent class are two columns. The first column displays the multiset class in long form, and the second displays the IC vector. The unison position in this seven-position vector falls before the slash. In traditional set theory this position, when

used at all, would simply display the cardinality of each set class because the constituents are considered through the identity function to be unisons "with themselves." This is acceptable when the representative pc in the class (the *type*) stands in for any and all of its members (the *tokens*). On the other hand, a multiset accounts for the precise number of tokens of each type. For this reason, the interval vector here only counts unisons when they exist between two or more tokens of the same type. That is, when there are pc duplications. This way, a multiset class of any cardinality contains the same number of entries in its vectors as does any traditional set classes of that cardinality. (Tetrachords, whether 0123 or 0011, contain six ics; pentachords contain 10, etc.) Also for this reason, there is such a multiset as 012 despite its lack of any pc duplication.

Multiplicity of interval classes in multiset functions can get quite high. Therefore IC vectors here contain letters representing interval totals according to the following system:

10 = a	36 = A	62 = !
11 = b	37 = B	63 = @
12 = c	38 = C	64 = #
13 = d	39 = D	65 = $
14 = e	40 = E	66 = %
15 = f	41 = F	
16 = g	42 = G	
17 = h	43 = H	
18 = i	44 = I	
19 = j	45 = J	
20 = k	46 = K	
21 = l	47 = L	
22 = m	48 = M	
23 = n	49 = N	
24 = o	50 = O	
25 = p	51 = P	
26 = q	52 = Q	
27 = r	53 = R	
28 = s	54 = S	
29 = t	55 = T	
30 = u	56 = U	
31 = v	57 = V	
32 = w	58 = W	
33 = x	59 = X	
34 = y	60 = Y	
35 = z	61 = Z	

As there may be confusion between the number **1** and the lowercase letter **l**, one should consider the total number of intervals for any cardinality and make sure the vector "adds up" appropriately.

Any parent class that is symmetrical by inversion or transposition (except by T_0) will contain fewer multiset offspring. For example, 012 may generate the tetrachords 0012, 0112, and 0122. Through inversion, 0012 and 0122 can be shown to belong to the same multiset class, and 0012 is the preferred representative of that class. Preference is given to the higher multiplicity of the first object, 0. In case of a tie, preference is the given to multiplicity of the second object, and so forth. This rule is becomes clearer when looking only at the multiplicity vectors (mvectors). Any trichord parent class, in order to produce a tetrachord offspring, can acquire the following mvectors: <2,1,1>, <1,2,1>, and <1,1,2>. Much like an interval adjacency series in a traditional set, the multiplicity function can help to identify symmetry in a multiset. So, for parent class 012, mfunctions <2,1,1> and <1,1,2> will produce members of the same multiset class. Preference is given to <2,1,1>. For parent class 027, a different pair shows symmetry. For this class, mfunctions <2,1,1> and <1,2,1> will produce inversionally related multisets 0027 and 0227. The preferred representative of the multiset class is 0027. Obviously the more degrees of symmetry a parent class has, the fewer multiset classes it can produce. In this table, wherever an mfunction redundantly produces and inversion or transposition of a more preferred multiset, an x is used as a place marker.

Although as a reference this document is hardly worth its size—calculating an IC vector is an easy task for music theorists—the massive amount of potential statistical data collected here is essential to further exploration of the properties of multisets.

APPENDIX 2

PROGRAM: *mfunctionmaker*

```
100 rem GETTING INITIAL DATA
200 input "number of objects? ",objects
210 objects = int(objects)
213 if objects < 1 then print "Uh, oh!  Too few" else goto 217
215 print
216 goto 200
217 if objects > 12 then print "Uh, oh!  Too many" else goto 220
218 print
219 goto 200
220 print
221 print "---- OK, " objects "objects (pcs)"
250 print
300 input "minimum number of elements? ",minelements
350 print
400 if minelements < objects then print "---- Uh, oh!  There must be at least as
many objects as elements!" else goto 600
410 print
500 goto 300
600 input "maximum number of elements? ",maxelements
650 if maxelements < minelements then print "---- That's illogical!" else goto 700
660 print
670 goto 600
700 rem SENDING to OBJECT SIZE
710 if objects = 1 then goto 1000
720 if objects = 2 then goto 2000
730 if objects = 3 then goto 3000
740 if objects = 4 then goto 4000
750 if objects = 5 then goto 5000
760 if objects = 6 then goto 6000
770 if objects = 7 then goto 7000
780 if objects = 8 then goto 8000
790 if objects = 9 then goto 9000
800 if objects = 10 then goto 10000
810 if objects = 11 then goto 11000
820 if objects = 12 then goto 12000
1000 rem **********ONE OBJECT**********
1005 print
1010 print "---- Sorry, we'll get to 'singletons' later!"
1020 end
2000 rem **********TWO OBJECTS**********
2005 print
2010 print "---- Sorry, we'll get to 'doubletons' later!"
2020 end
3000 rem **********THREE OBJECTS**********
3100 open "mfunc3.txt" for output as #3
```

```
3150 count = 0
3200 for a = 1 to maxelements-2
3210 for b = 1 to maxelements-2
3230 for c = 1 to maxelements-2
3400 if a+b+c > maxelements then 3600
3450 if a+b+c < minelements then 3600
3475 count = count+1
3500 print #3,a+b+c,",",a,",",b,",",c
3600 next c
3610 next b
3620 next a
3800 print
3805 print #3,0,0,0,0,0
3810 print "---- Done.  I found " count "mfunctions!"
3820 print "---- See the 'mfunc3.txt' file"
3830 print
3850 close #3
3900 end
4000 rem **********FOUR OBJECTS**********
4100 open "mfunc4.txt" for output as #4
4150 count = 0
4200 for a = 1 to maxelements-3
4210 for b = 1 to maxelements-3
4230 for c = 1 to maxelements-3
4240 for d = 1 to maxelements-3
4400 if a+b+c+d > maxelements then 4600
4450 if a+b+c+d < minelements then 4600
4475 count = count+1
4500 print #4,a+b+c+d,",",a,",",b,",",c,",",d
4600 next d
4610 next c
4620 next b
4630 next a
4800 print
4805 print #4,0,0,0,0,0,0
4810 print "---- Done.  I found " count "mfunctions!"
4820 print "---- See the 'mfunc4.txt' file"
4830 print
4850 close #4
4900 end
5000 rem **********FIVE OBJECTS**********
5100 open "mfunc5.txt" for output as #5
5150 count = 0
5200 for a = 1 to maxelements-4
5210 for b = 1 to maxelements-4
5230 for c = 1 to maxelements-4
5240 for d = 1 to maxelements-4
5250 for e = 1 to maxelements-4
5400 if a+b+c+d+e > maxelements then 5600
5450 if a+b+c+d+e < minelements then 5600
5475 count = count+1
5500 print #5,a+b+c+d+e,",",a,",",b,",",c,",",d,",",e
5600 next e
```

```
5610 next d
5620 next c
5630 next b
5640 next a
5800 print
5805 print #5,0,0,0,0,0,0,0
5810 print "---- Done.  I found " count "mfunctions!"
5820 print "---- See the 'mfunc5.txt' file"
5830 print
5850 close #5
5900 end
6000 rem **********SIX OBJECTS**********
6100 open "mfunc6.txt" for output as #6
6150 count = 0
6200 for a = 1 to maxelements-5
6210 for b = 1 to maxelements-5
6230 for c = 1 to maxelements-5
6240 for d = 1 to maxelements-5
6250 for e = 1 to maxelements-5
6260 for f = 1 to maxelements-5
6400 if a+b+c+d+e+f > maxelements then 6600
6450 if a+b+c+d+e+f < minelements then 6600
6475 count = count+1
6500 print #6,a+b+c+d+e+f,",",a,",",b,",",c,",",d,",",e,",",f
6600 next f
6610 next e
6620 next d
6630 next c
6640 next b
6650 next a
6800 print
6805 print #6,"0  ,0 ,0 ,0 ,0 ,0 ,0"
6810 print "---- Done.  I found " count "mfunctions!"
6820 print "---- See the 'mfunc6.txt' file"
6830 print
6850 close #6
6900 end
7000 rem **********SEVEN OBJECTS**********
7100 open "mfunc7.txt" for output as #7
7150 count = 0
7200 for a = 1 to maxelements-6
7210 for b = 1 to maxelements-6
7230 for c = 1 to maxelements-6
7240 for d = 1 to maxelements-6
7250 for e = 1 to maxelements-6
7260 for f = 1 to maxelements-6
7270 for g = 1 to maxelements-6
7400 if a+b+c+d+e+f+g > maxelements then 7600
7450 if a+b+c+d+e+f+g < minelements then 7600
7475 count = count+1
7500 print #7,a+b+c+d+e+f+g,",",a,",",b,",",c,",",d,",",e,",",f,",",g
7600 next g
7610 next f
```

```
7620 next e
7630 next d
7640 next c
7650 next b
7660 next a
7800 print
7805 print #7,"0  ,0 ,0 ,0 ,0 ,0 ,0 ,0"
7810 print "---- Done.  I found " count "mfunctions!"
7820 print "---- See the 'mfunc7.txt' file"
7830 print
7850 close #7
7900 end
8000 rem **********EIGHT OBJECTS**********
8100 open "mfunc8.txt" for output as #8
8150 count = 0
8200 for a = 1 to maxelements-7
8210 for b = 1 to maxelements-7
8230 for c = 1 to maxelements-7
8240 for d = 1 to maxelements-7
8250 for e = 1 to maxelements-7
8260 for f = 1 to maxelements-7
8270 for g = 1 to maxelements-7
8280 for h = 1 to maxelements-7
8400 if a+b+c+d+e+f+g+h > maxelements then 8600
8450 if a+b+c+d+e+f+g+h < minelements then 8600
8475 count = count+1
8500 print #8,a+b+c+d+e+f+g+h,",",a,",",b,",",c,",",d,",",e,",",f,",",g,",",h
8600 next h
8610 next g
8620 next f
8630 next e
8640 next d
8650 next c
8660 next b
8670 next a
8800 print
8805 print #8,"0  ,0 ,0 ,0 ,0 ,0 ,0 ,0"
8810 print "---- Done.  I found " count "mfunctions!"
8820 print "---- See the 'mfunc8.txt' file"
8830 print
8850 close #8
8900 end
9000 rem **********NINE OBJECTS**********
9100 open "mfunc9.txt" for output as #9
9150 count = 0
9200 for a = 1 to maxelements-8
9210 for b = 1 to maxelements-8
9230 for c = 1 to maxelements-8
9240 for d = 1 to maxelements-8
9250 for e = 1 to maxelements-8
9260 for f = 1 to maxelements-8
9270 for g = 1 to maxelements-8
9280 for h = 1 to maxelements-8
```

```
9290 for i = 1 to maxelements-8
9400 if a+b+c+d+e+f+g+h+i > maxelements then 9600
9450 if a+b+c+d+e+f+g+h+i < minelements then 9600
9475 count = count+1
9500 print #9,a+b+c+d+e+f+g+h+i,",",a,",",b,",",c,",",d,",",e,",",f,",",g,",",h,",",i
9600 next i
9610 next h
9620 next g
9630 next f
9640 next e
9650 next d
9660 next c
9670 next b
9680 next a
9800 print
9805 print #9,"0  ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0"
9810 print "---- Done.  I found " count "mfunctions!"
9820 print "---- See the 'mfunc9.txt' file"
9830 print
9850 close #9
9900 end
10000 rem **********TEN OBJECTS**********
10100 open "mfunc10.txt" for output as #1
10150 count = 0
10200 for a = 1 to maxelements-9
10210 for b = 1 to maxelements-9
10230 for c = 1 to maxelements-9
10240 for d = 1 to maxelements-9
10250 for e = 1 to maxelements-9
10260 for f = 1 to maxelements-9
10270 for g = 1 to maxelements-9
10280 for h = 1 to maxelements-9
10290 for i = 1 to maxelements-9
10300 for j = 1 to maxelements-9
10400 if a+b+c+d+e+f+g+h+i+j > maxelements then 10600
10450 if a+b+c+d+e+f+g+h+i+j < minelements then 10600
10475 count = count+1
10500 print
#10,a+b+c+d+e+f+g+h+i,",",a,",",b,",",c,",",d,",",e,",",f,",",g,",",h,",",i,",",j
10600 next j
10610 next i
10620 next h
10630 next g
10640 next f
10650 next e
10660 next d
10670 next c
10680 next b
10690 next a
10800 print
10805 print #1,"0  ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0"
10810 print "---- Done.  I found " count "mfunctions!"
10820 print "---- See the 'mfunc10.txt' file"
```

```
10830 print
10850 close #1
10900 end
11000 rem **********ELEVEN OBJECTS**********
11100 open "mfunc11.txt" for output as #1
11150 count = 0
11200 for a = 1 to maxelements-10
11210 for b = 1 to maxelements-10
11230 for c = 1 to maxelements-10
11240 for d = 1 to maxelements-10
11250 for e = 1 to maxelements-10
11260 for f = 1 to maxelements-10
11270 for g = 1 to maxelements-10
11280 for h = 1 to maxelements-10
11290 for i = 1 to maxelements-10
11300 for j = 1 to maxelements-10
11310 for k = 1 to maxelements-10
11400 if a+b+c+d+e+f+g+h+i+j+k > maxelements then 11600
11450 if a+b+c+d+e+f+g+h+i+j+k < minelements then 11600
11475 count = count+1
11500 print
#1,a+b+c+d+e+f+g+h+i+j+k,",",a,",",b,",",c,",",d,",",e,",",f,",",g,",",h,",",i,",",j,",",
k
11600 next k
11610 next j
11620 next i
11630 next h
11640 next g
11650 next f
11660 next e
11670 next d
11680 next c
11690 next b
11700 next a
11800 print
11805 print #1,"0  ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0"
11810 print "---- Done.  I found " count "mfunctions!"
11820 print "---- See the 'mfunc11.txt' file"
11830 print
11850 close #1
11900 end
12000 rem **********TWELVE OBJECTS**********
12100 open "mfunc12.txt" for output as #1
12150 count = 0
12200 for a = 1 to maxelements-11
12210 for b = 1 to maxelements-11
12230 for c = 1 to maxelements-11
12240 for d = 1 to maxelements-11
12250 for e = 1 to maxelements-11
12260 for f = 1 to maxelements-11
12270 for g = 1 to maxelements-11
12280 for h = 1 to maxelements-11
12290 for i = 1 to maxelements-11
```

```
12300 for j = 1 to maxelements-11
12310 for k = 1 to maxelements-11
12320 for l = 1 to maxelements-11
12400 if a+b+c+d+e+f+g+h+i+j+k+l > maxelements then 12600
12450 if a+b+c+d+e+f+g+h+i+j+k+l < minelements then 12600
12475 count = count+1
12500 print
#1,a+b+c+d+e+f+g+h+i+j+k+l,",",a,",",b,",",c,",",d,",",e,",",f,",",g,",",h,",",i,",",j,"
,",k,",",l
12600 next l
12610 next k
12620 next j
12630 next i
12640 next h
12650 next g
12660 next f
12670 next e
12680 next d
12690 next c
12700 next b
12710 next a
12800 print
12805 print #1,"0  ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0"
12810 print "---- Done.  I found " count "mfunctions!"
12820 print "---- See the 'mfunc12.txt' file"
12830 print
12850 close #1
12900 end
```

APPENDIX 3

PROGRAM: *vectormaker*

Only the *vectormaker* code for tetrachords is included here. The code for other cardinalities includes only minor variation to accommodate more objects.

```
50 rem INPUTING PITCH CLASSES**********************
100 input "pc 1? ",pca
110 if pca < 0 or pca > 11 then print "---------0-11 please!" else goto 200
120 goto 100

200 input "pc 2? ",pcb
210 if pcb < 0 or pcb > 11 then print "---------0-11 please!" else goto 220
215 goto 200
220 if pcb = pca then print "---------That pc is taken!" else goto 300
230 goto 200

300 input "pc 3? ",pcc
310 if pcc < 0 or pcc > 11 then print "---------0-11 please!" else goto 320
315 goto 300
320 if pcc = pca or pcc = pcb then print "---------That pc is taken!" else goto 400
330 goto 300

400 input "pc 4? ",pcd
410 if pcd < 0 or pcd > 11 then print "---------0-11 please!" else goto 420
415 goto 400
420 if pcd = pca or pcd = pcb or pcd=pcc then print "---------That pc is taken!" else
goto 500
430 goto 400

500 print "great"

2000 rem **********************FINDING
INTERVALS**************************
2100 if pcb-pca < 0 then ointa = pcb-pca+12 else ointa = pcb-pca
2110 if ointa > 6 then inta = 12-ointa else if ointa = 6 then inta = 6 else inta = ointa

2200 if pcc-pca < 0 then ointb = pcc-pca+12 else ointb = pcc-pca
2210 if ointb > 6 then intb = 12-ointb else if ointb = 6 then intb = 6 else intb =
ointb

2300 if pcd-pca < 0 then ointc = pcd-pca+12 else ointc = pcd-pca
2310 if ointc > 6 then intc = 12-ointc else if ointc = 6 then intc = 6 else intc = ointc

2400 if pcc-pcb < 0 then ointd = pcc-pcb+12 else ointd = pcc-pcb
```

```
2410 if ointd > 6 then intd = 12-ointd else if ointd = 6 then intd = 6 else intd =
ointd

2500 if pcd-pcb < 0 then ointe = pcd-pcb+12 else ointe = pcd-pcb
2510 if ointe > 6 then inte = 12-ointe else if ointe = 6 then inte = 6 else inte = ointe

2600 if pcd-pcc < 0 then ointf = pcd-pcc+12 else ointf = pcd-pcc
2610 if ointf > 6 then intf = 12-ointf else if ointf = 6 then intf = 6 else intf = ointf

3000 open "mfunction4_25e.txt" for input as #1
3005 open "tempICvector.txt" for output as #2
3009 input #1,elem,multa,multb,multc,multd
3011 iczeros = 0
3012 icones = 0
3013 ictwos = 0
3014 icthrees = 0
3015 icfours = 0
3016 icfives = 0
3017 icsixes = 0
3018 if elem = 0 then goto 6715

3020 facta = multa*multb
3030 factb = multa*multc
3040 factc = multa*multd
3050 factd = multb*multc
3060 facte = multb*multd
3070 factf = multc*multd
3100 iczeros = (multa*(multa-1)/2)+(multb*(multb-1)/2)+(multc*(multc-
1)/2)+(multd*(multd-1)/2)

4100 if inta = 1 then icones = facta
4110 if intb = 1 then icones = icones+factb
4120 if intc = 1 then icones = icones+factc
4130 if intd = 1 then icones = icones+factd
4140 if inte = 1 then icones = icones+facte
4150 if intf = 1 then icones = icones+factf

4200 if inta = 2 then ictwos = facta
4210 if intb = 2 then ictwos = ictwos+factb
4220 if intc = 2 then ictwos = ictwos+factc
4230 if intd = 2 then ictwos = ictwos+factd
4240 if inte = 2 then ictwos = ictwos+facte
4250 if intf = 2 then ictwos = ictwos+factf

4300 if inta = 3 then icthrees = facta
4310 if intb = 3 then icthrees = icthrees+factb
4320 if intc = 3 then icthrees = icthrees+factc
4330 if intd = 3 then icthrees = icthrees+factd
4340 if inte = 3 then icthrees = icthrees+facte
4350 if intf = 3 then icthrees = icthrees+factf

4400 if inta = 4 then icfours = facta
4410 if intb = 4 then icfours = icfours+factb
```

```
4420 if intc = 4 then icfours = icfours+factc
4430 if intd = 4 then icfours = icfours+factd
4440 if inte = 4 then icfours = icfours+facte
4450 if intf = 4 then icfours = icfours+factf

4500 if inta = 5 then icfives = facta
4510 if intb = 5 then icfives = icfives+factb
4520 if intc = 5 then icfives = icfives+factc
4530 if intd = 5 then icfives = icfives+factd
4540 if inte = 5 then icfives = icfives+facte
4550 if intf = 5 then icfives = icfives+factf

4600 if inta = 6 then icsixes = facta
4610 if intb = 6 then icsixes = icsixes+factb
4620 if intc = 6 then icsixes = icsixes+factc
4630 if intd = 6 then icsixes = icsixes+factd
4640 if inte = 6 then icsixes = icsixes+facte
4650 if intf = 6 then icsixes = icsixes+factf

6700 print
#2,elem;",";iczeros;",";icones;",";ictwos;",";icthrees;",";icfours;",";icfives;","
icsixes;",";multa;",";multb;",";multc;",";multd
6710 goto 3009
6715 print
#2,elem;",";iczeros;",";icones;",";ictwos;",";icthrees;",";icfours;",";icfives;",";icsixes
;",";0;",";0;",";0;",";0
6716 close #1
6717 close #2
6720 print "Done!  Go see tempICVector file for results"
6730 end
```

APPENDIX 4

PROGRAM: *vectorcomparer*

Several versions of *vectorcomparer* were written in order to compare vectors of the offspring of parent classes with varying numbers of objects and elements. Shown below is the version designed to compare a tetrachord parent class with a trichord parent class. Other modifications were made, creating variations that eliminate self-symmetrical offspring when comparing a parent class with itself. Those are not included here.

```
100 rem **************CHOOSING SET CLASSES *************
110 print "What's your tetrachord?"
120 print "Give pcs in prime form with no spaces and no commas."
130 input "Then add '.txt'  ",scX$
140 print
145 print "What's your trichord?"
150 print "Give pcs in prime form with no spaces and no commas."
160 input "Then add '.txt'  ",scY$
170 print


200 rem ********************OPENING LISTS***************
205 open scX$ for input as #1
210 input
#1,elemX,iczerosX,iconesX,ictwosX,icthreesX,icfoursX,icfivesX,icsixesX,multaX,multb
X,multcX,multdX
215 if elemX=0 then goto 510
220 open scY$ for input as #2
230 input
#2,elemY,iczerosY,iconesY,ictwosY,icthreesY,icfoursY,icfivesY,icsixesY,multaY,multbY
,multcY


300 rem ************************CHECKING ELEMENTS******************
310 if elemY<elemX then goto 230 else if elemY>elemX or elemY=0 then goto 325
else goto 420
325 close #2
330 goto 210


400 rem ********************COMPARING VECTORS*******************
420 print elemX;" elements in ";scX$;" and ";elemY;" elements in ";scY$
```

```
426 if iczerosX=iczerosY and iconesX=iconesY and ictwosX=ictwosY and
icthreesX=icthreesY and icfoursX=icfoursY and icfivesX=icfivesY and
icsixesX=icsixesY then goto 429 else goto 230
429 print
430 print scX$;" with mfunction ";"<";multaX;",";multbX;",";multcX;",";multdX;">"
440 print "and"
450 print scY$;" with mfunction ";"<";multaY;",";multbY;",";multcY;">"
460 print "have vector: ";iczerosX;" ";iconesX;" ";ictwosX;" ";icthreesX;" ";icfoursX;"
";icfivesX;" ";icsixesX
465 print
470 input "More? (y/n) ",more$
480 if more$="y" then goto 230 else if more$="n" then goto 495
490 goto 470

495 print "OK, we'll stop there."
496 goto 500
497 print "That's all!  I've checked every one!"
500 close #2
510 close #1
530 end
```

BIBLIOGRAPHY

Alegant, Brian. "When Even Becomes Odd: A Partitional Approach to Inversion." *Journal of Music Theory* 43/2 (Autumn, 1999): 193-230.

Babbitt, Milton. "Twelve-Tone Invariants as Compositional Determinants." *Musical Quarterly* 46/2 (1960): 246-59. Reprinted in *The Collected Essays of Milton Babbitt*, edited by Stephen Peles, Stephen Dembski, Andrew Mead, and Joseph N. Straus. Princeton, NJ: Princeton University Press, 2003, 55-69.

———. "Set Structure as Compositional Determinant." *Journal of Music Theory* 5/1 (1961): 72-94. Reprinted in *The Collected Essays of Milton Babbitt*, edited by Stephen Peles, Stephen Dembski, Andrew Mead, and Joseph N. Straus. Princeton, NJ: Princeton University Press, 2003, 86-108.

———. "The Structure and Function of Musical Theory." *College Music Symposium* 5 (Fall, 1965): 49-60. Reprinted in *The Collected Essays of Milton Babbitt*, edited by Stephen Peles, Stephen Dembski, Andrew Mead, and Joseph N. Straus. Princeton, NJ: Princeton University Press, 2003, 191-201.

———. "Remarks on the Recent Stravinsky." *Perspectives of New Music* 2/2 (1964): 35-55. Reprinted in *The Collected Essays of Milton Babbitt*, edited by Stephen Peles, Stephen Dembski, Andrew Mead, and Joseph N. Straus. Princeton, NJ: Princeton University Press, 2003, 147-171.

Blizard, Wayne. "Negative Membership." *Notre Dame Journal of Formal Logic* 31/3 (Summer, 1990): 346-368.

Bromberger, Sylvain. "Types and Tokens in Linguistics." In *On What We Know We Don't Know*. Chicago: University of Chicago, 1992.

Callender, Cliff, Ian Quinn, and Dmitri Tymozcko. "Generalized Voice-Leading Spaces." *Science* 320 (April 18, 2008): 346-48.

Callender, Clifton and Rachel Hall. "Crystallography and the Structure of Z-related Sets." Society for Music Theory Annual Meeting, Nashville, TN, November, 2008.

Cohn, Richard. *Transpositional Combination in Twentieth-Century Music*. Ph.D. dissertation, Eastman School of Music, University of Rochester, 1986.

———. "Inversional Symmetry and Transpositional Combination in Bartók." *Music Theory Spectrum* 10 (Spring, 1988): 19-42.

Cone, Edward T. "Schubert's Promissory Note: An Exercise in Musical Hermenuetics." *19th-Century Music* 5/3 (Spring, 1982): 233-241.

Forte, Allen. *The Structure of Atonal Music*. New Haven: Yale University Press, 1973.

Gamer, Carlton and Paul Lansky. "Fanfare for the Common Tone." *Perspectives of New Music* 14/2 (1976): 229-235.

Goodman, Nelson. *The Structure of Appearance,* 3rd ed. Dordrecht: D. Reidel, 1977.

Hale, Bob. *Abstract Objects.* New York: Basil Blackwell, 1987.

Hanson, Howard. *Harmonic Materials of Modern Music: Resources of the Tempered Scale*. New York: Appleton-Century-Crofts, 1960.

Herdan, G. *Type-token mathematics: a textbook of mathematical linguistics*. Mouton: The Hague, 1960.

Hook, Julian. "Why Are There Twenty-Nine Tetrachords?" *Music Theory Online* 13.4 (December 2007).

Merlini, Donatella, Francesca Uncini, and M. Cecilia Verri, "A Unified Approach to the Study of General and Palindromic Compositions." *Integers: Electronic Journal of Number Theory* 4 (2004), #A23.

Lewin, David. "On Extended Z-Triples." *Theory and Practice* 7/1 (August, 1982): 38-39.

———. "Transformational Techniques in Atonal and Other Music Theories." *Perspectives of New Music* 21 (1982-3): 312-71

Martino, Donald. "The Source Set and Its Aggregate Formations." *Journal of Music Theory* 5/2 (Winter, 1961): 224-73.

Morris, Robert D. *Composition With Pitch Classes.* New Haven: Yale University Press, 1983.

Morris, Robert D. *Class Notes for Atonal Music Theory*. Lebanon, N.H.: Frog Peak Music, 1991.

Morris, Robert D. "Set Groups, Complementation, and Mappings among Pitch-Class Sets." *Journal of Music Theory* 26/1 (Spring, 1982): 101-144.

Morris, Robert D. "Pitch-Class Duplication in Serial Music: Partitions of the Double Aggregate." *Perspectives of New Music* 41/2 (Summer 2003): 96-119.

O'Donnell, Shaugn. "Klumpenhouwer Networks, Isography, and the Molecular Metaphor." *Intégral* 12 (1998): 53–80.

———. "Embracing Relational Abundance." *Music Theory Online* 13/3 (September 2007).

Peirce C.S. *Collected Papers of Charles S. Peirce*, Vol. IV The Simplest Mathematics, edited by Charles Hart Shorne and Paul Weiss. Cambridge, MA: Harvard University Press, 1933.

Quine, W.V. and Nelson Goodman. "Steps Toward a Constructive Nominalism." In Nelson Goodman. *Problems and Projects.* New York: Bobbs-Merrill, 1972.

Quine, W.V. *Quiddities: An Intermittently Philosophical Dictionary*. Cambridge, MA: Belknap, 1987.

Quinn, Ian. *A Unified Theory of Chord Quality in Equal Temperaments*. Ph.D. dissertation, Eastman School of Music, University of Rochester, 2004.

———. "General Equal-Tempered Harmony, Part One" *Perspectives of New Music* 44/2 (Summer, 2006).

———. "General Equal-Tempered Harmony, Part Two" *Perspectives of New Music* 45/1 (Winter, 2007).

Rahn, John. *Basic Atonal Theory*. New York: Longman, 1980.

Révész, Géza. *Einführung in die Musikpsychologie*. Berne, 1946. Translated as *Introduction to the Psychology of Music*. Norman: University of Oklahoma, 1953.

Robinson, Thomas. "Pitch-Class-Balance Space." Graduate Students in Music Symposium, The Graduate Center, CUNY, New York, NY, April, 2006.

———. "The End of Similarity? Semitonal Offset as Similarity Measure." Paper presented at the annual meeting of the Music Theory Society of New York State, Saratoga Springs, NY, April, 2006.

Soderberg, Stephen. "Z-Related Sets as Dual Inversions." *Journal of Music Theory* 39/1 (Spring, 1995): 77-100.

Straus, Joseph N. *Introduction to Post-Tonal Theory*, 3rd ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2005.

Tymozcko, Dmitri. "The Geometry of Musical Chords." *Science* 313 (July 7, 2006): 72-74.

Wetzel, Linda. "What Are Occurrences of Expressions?" *Journal of Philosophical Logic* 22 (1993): 215-220.

———. "On Types and Words." *Journal of Philosophical Research* 27 (2002): 240.

———. "The Trouble with Nominalism." *Philosophical Studies* 98 (2000): 361-70.

———. *Types and Tokens: An Essay on Universals*. Cambridge, MA: The M.I.T. Press, 2008.

Wollheim, Richard. *Art and Its Objects,* 2nd ed. Cambridge: Cambridge University Press, 1980[1968].

Wolterstorff, Nicholas. *Worlds and Works of Art*. Oxford: Clarendon Press, 1980.

Ziff, Paul. "What is Said." In *Semantics of Natural Language,* 2nd ed., edited by Donald Davidson and Gilbert Harman. Dordrecht, Holland: D. Reidel, 1972.