Mathematics and Society
University of Isfahan
http://math-sci.ui.ac.ir

Journal of Mathematics and Society
ISSN (print): 2345-6493, ISSN (on-line): 2345-6507
Vol. 10 No. 2 (2025), pp. 115-162.

http://www.ui.ac.ir

A GAP TUTORIAL FOR TRANSFORMATIONAL MUSIC THEORY**

ROBERT W. PECK
TRANSLATORS: MEHDI SHAMS* AND SEYYED ALI MOHAMMADIYEH

Abstract. GAP (an acronym for Groups, Algorithms, and Programming) is a computational discrete algebra system that can be applied as a practical tool for transformational music theory. In particular, this software offers theorists tools that are not easily accessible elsewhere. This tutorial demonstrates how to use GAP to generate algebraic group structures utilized by music theorists. Additionally, it explores how these structures can be applied to the study of transformational theory using familiar concepts derived from the Klumpenhouwer network and Neo-Riemannian theories.

1. Introduction

GAP (an acronym for Groups, Algorithms, and Programming) is a mathematical software package that has great potential as a tool for researchers and students of transformational music theory. Whereas many computer resources are available to music theorists in the investigation of pitch-class set theory and serial theory—pitch-class set calculators, matrix generators, and the like—few, if any, permit users to construct, analyze, and manipulate algebraic systems themselves. Such techniques, however, are an increasingly large part of the transformation theorist's methodologies. It is often necessary to consider concepts such as the subgroup structure of a group, or to determine a group's

Keywords: computer-assisted research, transformational music theory, group theory, Klumpenhouwer networks, Neo-Riemannian theory
Article Type: Translation Paper.
Communicated by Alireza Abdollahi.
*Corresponding author.
Received: 12-11-2023, Accepted: 16-04-2025, Published Online: 30-06-2025.
** The above abstract has been extracted by the translator from the original article (R. W. Peck, A GAP tutorial for transformational music theory, *Music Theory Online*, **18** no. 2 (2011) pp. 22.).
http://dx.doi.org/10.22108/msci.2025.139767.1621 .

automorphisms. Whereas one can do such calculations by hand, the speed at which GAP performs them can dramatically increase efficiency of time for research. From a pedagogical point of view, GAP provides students of transformation theory a resourceful and effective environment for experimentation, and a means of providing preliminary empirical proofs for their results.

GAP was developed by and for mathematicians and other computational scientists. It began in 1986 as a project by four students at *RWTH* Aachen University, in Aachen, North Rhine-Westphalia, Germany: Johannes Meier, Alice Niemeyer, Werner Nickel, and Martin Schönert. Its first public release was in 1988, with version 2.4. Since that time, it has undergone a number of revisions, and has assumed worldwide recognition, (1) although it has not received much attention in musictheoretical research. (2) GAP 4.4.12 is the latest version. It is free and is available for Windows, Macintosh, and UNIX operating systems, along with all its documentation, at https://www.gap-system.org. An interactive online version is also available via Sage.

2. **Main Results**

Throughout this tutorial we use certain orthographic (i.e., notational) conventions, which sometimes differ from those commonly used in music theory. In some instances, these conventions are required by GAP; at other times, they are not necessary, but are more consistent with standard mathematical conventions. For instance, although it is not required by GAP, we represent all operations with lowercase letters: $g \coloneqq (1,2,3)$, $h \coloneqq (1,2)$.

This notation differs from the common practice in music theory of using uppercase letters for certain operators, such as T_n and I. For these particular operators, we instead use t^n and i, respectively. Operations also appear in cyclic notation. In the case of g above, the parenthetical expression $(1,2,3)$ tells us that, under g, the integer 1 moves to 2, 2 moves to 3, and 3 moves back to 1. The result is a permutation of order 3 (i.e., a 3-cycle). In contrast, under h, 1 moves to 2, and 2 moves directly back to 1 (i.e., a 2-cycle, or involution). Then, the composition of g and h, $gh = (1,2,3)(1,2)$, sends 1 to 2 under g, but then 2 back to 1 under h. Hence, it fixes (or stabilizes) 1. Further, it sends 2 to 3 under g, where it remains under h; and it sends 3 to 1 under g, and 1 to 2 under h. Overall, it stabilizes 1, sends 2 to 3, and 3 to 2, so $gh = (2,3)$.

We use multiplicative notation rather than additive notation, even for abelian (i.e., commutative) groups: $g \times h$, or gh (rather than $g + h$). Furthermore, GAP uses right-functional orthography (i.e., read from right to left). That is, in the composition gh, perform g first, then h. This practice differs from the left-functional notation commonly used in music theory, where, for instance, the operation $T_n I$ means invert first (I), then transpose by n (T_n).

We represent sets, groups, and other algebraic structures with uppercase letters: $G \coloneqq \langle g, h \rangle$. Here, the angle brackets signify that G is the group generated by g and h. Additionally, we give functions

 http://dx.doi.org/10.22108/msci.2025.139767.1621

that map such structures to one another (morphisms) with Greek letters; but as GAP reads only ASCII characters, we spell out these Greek letters: $pi : G^{pi} \to H$, where $g^{pi} \times h^{pi} = (g \times h)^{pi}$, for all $g, h \in G$.

In GAP, we use exponential notation for functions, as in the example above, rather than prefix notation, which is more common in music-theoretical writing. In other words, we write G^{pi}, rather than $pi(G)$. In this case, G^{pi} indicates a function pi that maps G to H, where g^{pi} is the image of g in the mapping, h^{pi} is the image of h, and so on. Such functions may be homomorphisms, as in this example; isomorphisms, which are one-to-one; surjective homomophisms; or automorphisms, which are isomorphisms of a group to itself.

Right-functional orthography is also consistent with GAP's use of exponential notation for functions. The product of functions *alpha* and *beta* (in that order) applied to a structure G receives the exponential notation:

$$\left(G^{alpha}\right)^{beta} = G^{alpha.beta}$$

whereas in prefix notation we would have:

$$beta\,(alpha\,(G)) = beta.alpha\,(G)$$

reading from the right to the left.

In cases where gh indicates a composition of group elements that act on a set S, or on a member x of that set, we once again use exponential notation, such as Sg^h, or xg^h, rather than prefix notation, such as $h(g\,(S))$ or $h\,(g\,(x))$. Finally, because we use multiplicative notation, we also use exponential notation for exponents in the usual sense (powers):

$$g \times g = g^2, \quad g \times g \times g = g^3, \quad \ldots$$

Throughout this tutorial, GAP commands are given in red, and GAP's corresponding output appear in blue. All commands are entered at the GAP prompt *gap >* (see Figure 1) and end in a semicolon (followed by [*enter*]). GAP commands may be typed with or without spaces. In GAP, we may define various objects using the standard notation :=" for definition. These objects include operations:

```
gap > g := (1,2,3);
(1,2,3)
gap > h := (1,2);
(1,2)
```

GAP places the output on the line(s) below the defining commands. In this case, the output confirms the definition by redisplaying the cyclic permutations that g and h represent.

FIGURE 1. The GAP welcome screen

We may similarly define algebraic structures:

$gap >\ \ G\ :=\ Group(g, h);$

```
Group([ (1,2,3), (1,2) ])
```

$gap >\ \ N\ :=\ Subgroup(G, [g]);$

```
Group([ (1,2,3) ])
```

Again, the output confirms the structure that is being defined: the group generated by g and h in the first case, and the subgroup of G generated by g in the second. Notice, however, that the output alone of the second command does not indicate that N is the subgroup of G generated by g.

We may also call objects from one of GAP's libraries:

$gap >\ \ D\_6\ :=\ DihedralGroup(6);$

```
<pc group of size 6 with 2 generators>
```

Here, the dihedral group of order 6 is a purely abstract algebraic structure, not a permutation group such as G above. The output confirms the size of the group, tells how many generators GAP uses to build the group, and references a particular type of abstract structure: a "*pcgroup*." The term "*pc*" in the output does not abbreviate "$pitch-class$", as appears commonly in music theory. Rather, it stands for "*polycyclic*", a type of group structure that includes the dihedral groups. Then, the elements of the D are not permutations, such as $(1, 2, 3)$ and $(1, 2)$, but rather compositions of abstract generators $f1$ (of order 2) and $f2$ (of order 3), as seen in the output of the following command.

We may define morphisms and other mappings:

$gap >\ \ pi\ :=\ IsomorphismGroups(D\_6, G);$

```
[ f1, f2 ] -> [ (2,3), (1,2,3) ]
```

The output for this command tells us that GAP has located an isomorphism from D_6 to G; otherwise, the output would indicate fail. Figure 2 provides a mapping for all six elements in each of the respective groups.

http://dx.doi.org/10.22108/msci.2025.139767.1621

$$f_1 \mapsto (2,3)$$
$$f_1 \cdot f_1 \mapsto (2,3)(2,3) = ()$$
$$f_2 \mapsto (1,2,3)$$
$$f_2 \cdot f_2 \mapsto (1,2,3)(1,2,3) = (1,3,2)$$
$$f_1 \cdot f_2 \mapsto (2,3)(1,2,3) = (1,2)$$
$$f_2 \cdot f_1 \mapsto (1,2,3)(2,3) = (3,1)$$

FIGURE 2. Mapping of D_6 to G under pi

We may display objects, including those defined previously:

```
gap> g;
(1,2,3)
gap> List(G);
[ (), (1,3,2), (1,2,3), (2,3), (1,3), (1,2) ]
```

Whereas the output to the "g;" command above displays a single cyclic permutation, the output for "$List(G)$" presents all six group elements in G. The empty parentheses, ( ), in the output that follows the latter command denotes the identity element of the group—for example, the trivial permutation, which sends each member of the set on which G acts (i.e., the integers 1, 2, and 3) to itself.

Additionally, we may determine products, factors, quotients, etc.:

```
gap> g * h;
(2,3)
gap> g ^ 2;
(1,3,2)
gap> DirectProduct(G, D_6);
<group of size 36 with 4 generators>
gap> G / N;
Group([ f1 ])
```

In GAP, we may perform various functions, arithmetic and otherwise:

```
gap> 1 * 2 * 3;
6
gap> Size(G);
6
```

and run various tests:

```
gap> Size(G) = Factorial(3);
true
gap> g * h = h * g;
false
gap> IsAbelian(G);
false
```

We may enter two or more commands on a single line (each followed by a semicolon), and GAP displays their respective output on adjacent successive lines:

```
gap> H := Subgroup(G, [h]); Size(H);
Group([ (1,2) ])
2
```

Long commands may be broken by [enter] before reaching the semicolon. In these instances, GAP continues the command line on a subsequent line with a simple "$>$" prompt:

```
gap> Size(InnerAutomorphismsAutomorphismGroup( [enter]
> AutomorphismGroup(G)));
1
```

The syntax of the above command is complex, but follows a certain logic. In a series of parentheticals, we are asking GAP first to establish the automorphism group for G ($AutomorphismGroup(G)$); second, to determine which of the automorphisms in this group are inner, hence derive from conjugation by a group element ($InnerAutomorphismsAutomorphismGroup$); and finally, to display how many of these inner automorphisms exist ($Size$). The output tells us that the inner automorphism group for G contains only one member.

We may suppress output for a command that would normally produce it by following the command with two semicolons:

```
gap> List(H);;
```

GAP stores a history of command lines for any one session. Users may recall a previous command by using the $\uparrow$ (up-arrow) key repeatedly as needed, and/or with the $\downarrow$ (down-arrow) key, to scroll through the history. Finally, to end a session, we use:

```
gap> quit;
```

Help for GAP users is available in several ways. First, the GAP website offers a number of online manuals, including tutorials, FAQ, examples, and instructions for joining the GAP Forum e-mail distribution

 http://dx.doi.org/10.22108/msci.2025.139767.1621

list. Help is also available within a GAP session; one may find help on a particular topic by typing a search term preceded by a question mark (again, the semicolon is not used):

```
gap > ?orbit
Help: several entries match this topic - type ?2 to get match [2]
[1] Tutorial: Orbit
[2] Reference: Orbit
[3] Reference: Orbit Stabilizer Methods for Polycyclic Groups
[4] Reference: Orbits
[5] Reference: Orbits!operation/attribute
[6] Reference: OrbitsDomain
[7] Reference: OrbitLength
[8] Reference: OrbitLengths
[9] Reference: OrbitLengthsDomain
[10] Reference: OrbitStabilizer
[11] Reference: OrbitStabilizerAlgorithm
[12] Reference: OrbitPerms
[13] Reference: OrbitsPerms
[14] Reference: OrbitStabChain
[15] Reference: OrbitPowerMaps
[16] Reference: OrbitFusions
[17] Extending: Orbits
[18] Extending: OrbitsishOperation
[19] Extending: OrbitishFO
[20] New Features: OrbitGenerators
[21] New Features: OrbitGeneratorsInv
[22] New Features: OrbitGeneratorsOfGroup
```

At a subsequent GAP prompt (which need not be immediate), we may enter the number of the entry we wish to read, following a question mark:

```
gap > ?2
<Orbit( <G>[, <Omega>], <pnt>, [<gens>, <acts>, ] <act>)
The orbit of the point <pnt>is the list of all images of <pnt>under the action.
(Note that the arrangement of points in this list is not defined by the
operation.)
The orbit of <pnt>will always contain one element that is *equal* to <pnt>,
```

```
however for performance reasons this element is not necessarily *identical* to
<pnt>, in particular if <pnt>is mutable.
```

$gap >$ $g := Group((1,3,2),(2,4,3));$

$gap >$ $Orbit(g,1);$

```
[ 1, 3, 2, 4 ]
```

$gap >$ $Orbit(g,[1,2],OnSets);$

```
[ [ 1, 2 ], [ 1, 3 ], [ 1, 4 ], [ 2, 3 ], [ 3, 4 ], [ 2, 4 ] ]
(See Section "Basic Actions" for information about specific actions.)
> Orbits( <G>, <seeds>[, <gens>, <acts>][, <act>] )!

{operation/attribute}
> Orbits( <xset>)!{operation/attribute}

--- <space> page, <n>next line, <b> back, <p> back line, <q> quit --
```

GAP then prompts for some form of continuation or exit from the manual; for example, n or q. The $[space]$, n, etc., are not followed by a semicolon (nor by $[enter]$). On encountering an error, the output provides a description of the problem, whether it is syntactical:

$gap >$ $Size(Group(g);$

```
Syntax error: ) expected
Size(Group(g);
^
```

or logical:

$gap >$ $G/H;$

```
Error, <N> must be a normal subgroup of <G> called from
<function>( <arguments>) called from read-eval-loop
Entering break read-eval-print loop ...
you can 'quit;' to quit to outer loop, or
you can 'return;' to continue
brk>
```

In the latter instance, we are given the opportunity to exit the description by typing

$brk >$ $quit;$

at the break prompt, or by continuing the description:

$brk >$ $return;$

```
Error, <H> must be contained in <G> called from
```

 http://dx.doi.org/10.22108/msci.2025.139767.1621

```
oper( super, sub ) called from
Index( G, N ) called from
oper( super, sub ) called from
NaturalHomomorphismByNormalSubgroupNCOrig( G, N ) called from
NaturalHomomorphismByNormalSubgroupNC( G, N ) called from
...
Entering break read-eval-print loop ...
you can 'quit;' to quit to outer loop, or
you can 'return;' to continue
brk>
```

Entering either *brk > return;* return; or *brk > quit;* at this point continues or quits the description accordingly.

Users may also access GAP on the World Wide Web via the mathematics software system Sage: www.sagemath.org. Including GAP, Sage has interfaces for Mathematica, Magma, and several other mathematics applications useful to music theorists. Therefore, in addition to computational discrete algebra, Sage may be used to study other algebras, calculus, number theory, cryptography, numerical computation, combinatorics, graph theory, and the like.

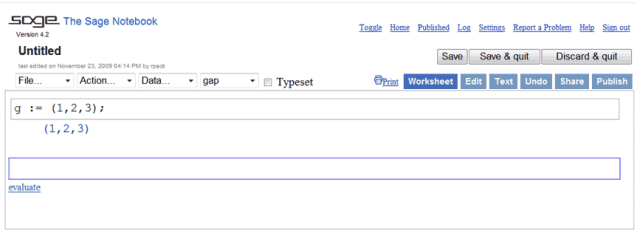

FIGURE 3. A sample Sage Notebook

To work in an online GAP session in Sage, create a notebook (or edit a pre-existing notebook). First, go to www.sagenb.org. Then, either sign in to an existing account, or register for a new one and sign in. For a new notebook, click on [*newworksheet*] and title it. This takes you to a worksheet (see Figure 3). From the drop-down menu near the top, select gap sage is the default). Type a GAP command into the first cell, and click on the corresponding [*evaluate*] link to display its output. A new cell appears, in which you may type the next command, and so on. You may save your worksheet (by clicking the [*save*] button), and publish your worksheet (by clicking the [*publish*] button). The latter gives you aURL, through which others may view your notebook. For example, a notebook $GAP - MTO$ which contains all of the commands in this tutorial, may be found at http://ssagenb.org/pub/1117/ .

Now let us build some familiar music-theoretical groups using GAP. In this section, we generate the transposition group T, the transposition and inversion group T/I, and the affine group TTO that includes the transpositions, inversions, and various multiplicative operations. We also examine how GAP can model these groups' actions on pitch classes and pitch-class sets, and how they relate to various abstract algebraic structures.

Call T the musical transposition group (mathematically, a translation group). T has an action on the set PCS of twelve pitch classes (residue classes of the infinite set of chromatic pitches under enharmonic and octave equivalence). We may use GAP to model groups such as T , using a permutation representation on some subset of the positive integers. Such representations in GAP do not incorporate the integer 0, to which the pitch class C is usually mapped. Instead, we map pitch class C to the integer 12. Put

$$\begin{aligned} C\# &= 1 \\ D &= 2 \\ E\flat &= 3 \\ &\vdots \\ B &= 11 \\ C &= 12 \end{aligned}$$

Hence, we may define a unit transposition t (translation), a pitch-class transposition "up" one semitone:

```
gap> t := (1,2,3,4,5,6,7,8,9,10,11,12);
(1,2,3,4,5,6,7,8,9,10,11,12)
```

We may compose operators using multiplication, such as t with t ((T_1 with (T_1):

```
gap> t * t;
(1,3,5,7,9,11)(2,4,6,8,10,12)
```

We may also use exponents for an operator composed with itself, as before:

```
gap> t^2;
(1,3,5,7,9,11)(2,4,6,8,10,12)
```

We may show inverses:

```
gap> t^-1;
(1,12,11,10,9,8,7,6,5,4,3,2)
gap> (t^2)^-1;
(1,11,9,7,5,3)(2,12,10,8,6,4)
```

 http://dx.doi.org/10.22108/msci.2025.139767.1621

We can use t as a generator of a group, `T`:

```
gap > T := Group(t);
Group([ (1,2,3,4,5,6,7,8,9,10,11,12) ])
```

The output for this command confirms that `T` is the group generated by t, where t is the permutation $(1,2,3,4,5,6,7,8,9,10,11,12)$. `T` is isomorphic to the cyclic group of order 12, C_{12}:

```
gap > IsomorphismGroups(T, CyclicGroup(12));
[ (1,5,9)(2,6,10)(3,7,11)(4,8,12), (1,4,7,10)(2,5,8,11)(3,6,9,12) ] ->
[ f3, f1*f2 ]
```

`GAP` can determine the orbit of a pitch class under the members of the `T` group, shown here using pitch class 1:

```
gap > Orbit(T, 1);
[ 1, 9, 5, 11, 7, 3, 12, 8, 4, 10, 6, 2 ]
```

`GAP` can also display the orbit of an unordered set of pitch classes (a pcset), such as a C major triad $\{0,4,7\}$ (recalling that we use the integer 12 in place of pitch class 0):

```
gap > Orbit(T, [4,7,12], OnSets);
[ [ 4, 7, 12 ], [ 4, 8, 11 ], [ 3, 8, 12 ], [ 2, 6, 9 ], [ 1, 6, 10 ],
[ 2, 5, 10 ], [ 1, 5, 8 ], [ 5, 9, 12 ], [ 1, 4, 9 ], [ 3, 7, 10 ],
[ 2, 7, 11 ], [ 3, 6, 11 ] ]
```

One way that we might use `GAP` as a tool for music theory is in the study of Klumpenhouwer networks (Lewin [17], Klumpenhouwer [11]). Klumpenhouwer networks, or $K-nets$, are directed graphs with nodes (vertices) labeled in the set of pitch classes and arrows (edges) in the `T/I` group (see Figure 4):

```
gap > t^2; i*t^3; i^5;
(1,3,5,7,9,11)(2,4,6,8,10,12)
(1,2)(3,12)(4,11)(5,10)(6,9)(7,8)
(1,4)(2,3)(5,12)(6,11)(7,10)(8,9)
```

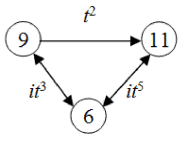

FIGURE 4. A sample $K-net$

Following Philip Lambert's ([14]) analysis, Figure 5 presents the first six measures of Arnold Schoenberg's Sechs kleine Klavierstücke, op. 19, no. 6.

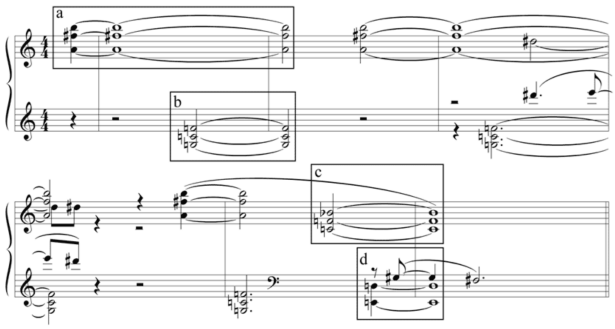

FIGURE 5. Schoenberg, Sechs kleine Klavierstücke, op. 19, no. 6, measures $1-6$ (following Lambert [14])

Another area of music-theoretic research for which GAP is useful is neo-Riemannian theory. This theory deals largely with contextual operations on the set of consonant triads, or Klänge. To that end, define an arbitrary consonant (major or minor) triad as an unordered pitch-class set (pcset):

```
gap> CM := [4,7,12];
[ 4, 7, 12 ]
```

Call the orbit of a pitch-class set under T/I a set-class (in this case, the set of pcsets to which a C major triad maps under the elements of the T/I group), and label the set-class of consonant triads K:

```
gap> K := Orbit(TI,CM,OnSets);
[ [ 4, 7, 12 ], [ 1, 5, 8 ], [ 5, 8, 12 ], [ 2, 6, 9 ], [ 4, 7, 11 ],
[ 1, 6, 9 ], [ 3, 7, 10 ], [ 3, 6, 10 ], [ 2, 7, 10 ], [ 3, 6, 11 ],
[ 4, 8, 11 ], [ 2, 5, 9 ], [ 3, 8, 11 ], [ 2, 5, 10 ], [ 5, 9, 12 ],
[ 1, 4, 8 ], [ 4, 9, 12 ], [ 1, 4, 9 ], [ 1, 6, 10 ], [ 3, 7, 12 ],
[ 1, 5, 10 ], [ 3, 8, 12 ], [ 2, 7, 11 ], [ 2, 6, 11 ] ]
```

K, as a set, has twenty-four members (twelve major triads and twelve minor triads, all represented above) on which T/I acts:

```
gap> TI_K := Action(TI,K,OnSets);
Group([ (1,2,4,7,11,15,19,23,22,18,14,10) (3,6,9,13,17,21,24,20,16,12,8,5),
(1,3)(2,5)(4,8)(6,10)(7,12)(9,14)(11,16) (13,18)(15,20)(17,22)(19,24)(21,23) ])
```

We can arrive at an isomorphic structure on $[1\ldots24]$ that is perhaps more intuitive. Let $[1\cdots12]$ represent the pitch classes that function as the roots of the major triads, and let $[13\cdots24]$ represent the roots of the minor triads, where x and $x+12$ belong to the same pitch class. (That is, 1 and 13 both map to the pitch class $C\#$; 2 and 14 to D; $\ldots$, 12 and 24 to C.) See Table 1.

http://dx.doi.org/10.22108/msci.2025.139767.1621

TABLE 1. Alternate mapping of K to $[1\cdots24]$

13	$\mapsto$	C#-	1	$\mapsto$	C#+
14	$\mapsto$	D-	2	$\mapsto$	D+
15	$\mapsto$	Eb-	3	$\mapsto$	Eb+
16	$\mapsto$	E-	4	$\mapsto$	E+
17	$\mapsto$	F-	5	$\mapsto$	F+
18	$\mapsto$	F#-	6	$\mapsto$	F#+
19	$\mapsto$	G-	7	$\mapsto$	G+
20	$\mapsto$	Ab-	8	$\mapsto$	Ab+
21	$\mapsto$	A-	9	$\mapsto$	A+
22	$\mapsto$	Bb-	10	$\mapsto$	Bb+
23	$\mapsto$	B-	11	$\mapsto$	B+
24	$\mapsto$	C-	12	$\mapsto$	C+

3. **Conclusions**

This tutorial endeavors to introduce some of the utilities in `GAP` to the music-theoretic community in the context of familiar transformation-theoretical concepts: transformation groups and their actions, Klumpenhouwer networks, and neo-Riemannian theory. Therefore, it presupposes a basic—but not advanced—understanding of these concepts, and no prior knowledge of `GAP`. After an initial section that describes the basics of the `GAP` environment (how to input commands, interpret output, get help, etc.), we generate the transposition group `T`, the transposition and inversion group `T/I`, and the affine group `TTO`. We examine how these groups act on the set of pitch classes and on sets of pitch-class sets, and perform various tests that relate them to well-known abstract-algebraic structures. Next, we use `GAP` to study Klumpenhouwer networks, generating the automorphism groups from which K-net isographies derive. Finally, we consider the Schritt and Wechsel group of neo-Riemannian theory, how it relates to the transposition and inversion group, how these are subgroups of Hook's [9] Uniform (and Quasi-uniform) Triadic Transformations, and how to generate other related groups of contextual operations.

The preceding discussion is intended to demonstrate `GAP`'s usefulness in studying transformational music theory. Whereas most of the examples recreate well known models, it has not been our purpose here to discover novel results. Nevertheless, `GAP` may function quite effectively in the pursuit of new theoretical systems. Moreover, it can be a valuable tool in transformation theory pedagogy, affording teachers and students an environment for experimentation with either existing or original concepts.

References

[1] M. Aschbacher, *Finite Group Theory*, Cambridge: Cambridge University Press, 1986.

[2] C. Callender, I. Quinn and D. Tymoczko, Generalized Voice Leading Spaces, *Science*, **320** (2008) 346–348.

[3] J. Clough, A Rudimentary Geometric Model for Contextual Transposition and Inversion, *J. Music Theory*, **42(2)** (1998) 297–306.

[4] R. Cohn, An Introduction to Neo-Riemannian Theory: A Survey and Historical Perspective, *J. Music Theory*, **42(2)** (1998) 167–180.

[5] R. Cohn, Uncanny Resemblances: Tonal Signification in the Freudian Age, *J. A. Music Sos.*, **57(2)** (2004) 285–323.

[6] J. D. Dixon and B. Mortimer, *Permutation Groups*, New York: Springer-Verlag, 1996.

[7] J. Douthett and R. Peck, An Order 1152 Group of Triadic Transformations and Its Relevance to Existing Musical Theoretical Structures, Paper presented at the Special Session on Mathematical Techniques in Musical Analysis, American Mathematical Society/Mathematical Association of America Joint National Meeting, New Orleans, Louisiana, 2007.

[8] The GAP Group, *GAP—Groups, Algorithms, and Programming, Version 4.4.12*, 2008. http://www.gap-system.org

[9] J. Hook Uniform Triadic Transformations, *J. Music Theory*, **46(1–2)** (2002) 57–126.

[10] B. Hyer, Reimag(in)ing Riemann, *J. Music Theory*, **39(1)** (1995) 101–138.

[11] H. Klumpenhouwer, *A Generalized Model of Voice-Leading for Atonal Music*, Ph.D. diss., Harvard University, 1991.

[12] H. Klumpenhouwer, The Inner and Outer Automorphisms of Pitch-Class Inversion and Transposition: Some Implications for Analysis with Klumpenhouwer Networks. *Intègral*, **12** (1998) 81–93.

[13] J. H. Kochavi, *Contextually Defined Musical Transformations*, Ph.D. diss., State University of New York, Buffalo, 2002.

[14] P. Lambert, Isographies and Some Klumpenhouwer Networks They Involve, *Music Theor. Spectrum*, **24(2)** (2002) 165–195.

[15] D. Lewin, Amfortas's Prayer to Titurel and the Role of D in Parsifal: The Tonal Spaces of the Drama and the Enharmonic C-flat/B, *Ninet-Cent. Music Rev.* **8(3)**, (1984) 336–349.

[16] D. Lewin, *Generalized Musical Intervals and Transformations*, New Haven: Yale University Press, 1987.

[17] D. Lewin, Klumpenhouwer Networks and Some Isographies that Involve Them, *Music Theor. Spectrum*, **12(1)** (1990) 83–120.

[18] D. Lewin, A Tutorial on Klumpenhouwer Networks, Using the Chorale in Schoenberg's Opus 11, No. 2, *J. Music Theory*, **38(1)** (1994) 79–101.

[19] R. D. Morris, Set Groups, Complementation, and Mappings among Pitch-Class Sets, *J. Music Theory*, **26(1)** (1982) 101–144.

[20] R. D. Morris, *Composition with Pitch-Classes*, New Haven: Yale University Press, 1987.

 http://dx.doi.org/10.22108/msci.2025.139767.1621

[21] R. Peck, *GAP (Groups, Algorithms, and Programming): A Tool for Computer-Assisted Research in Music Theory*, Poster presented at Society for Music Theory National Meeting, Cambridge, Massachusetts, 2005.

[22] R. Peck, Wreath Products in Transformational Music Theory, *Perspect. New Music*, **47(1)** (2009) 193–210.

[23] D. Smyth, More About Wagner's Chromatic Magic, *31st Annual Meeting of Society for Music Theory*, Nashville, Tennessee, 2008.

[24] D. V. Starr, Sets, Invariance, and Partitions, *J. Music Theory*, **22(1)** (1978) 136–183.

[25] D. Starr and Robert Morris, The Structure of All-Interval Series, *J. Music Theory*, **18(2)** (1974) 364–389.

[26] D. Starr and R. Morris. A General Theory of Combinatoriality and the Aggregate, *Perspect. New Music*, **16(1)** 3–35 (1977); and **16(2)** (1977) 50–84.

[27] W. Stein, et al. *Sage Mathematics Software, Version 3.4*, The Sage Development Team, 2009. http://www.sagemath.org/

Mehdi Shams

Department of Statistics, Faculty of Mathematical Sciences, University of Kashan, Kashan 87317-53153 Kashan, I. R. Iran

Email: mehdishams@kashanu.ac.ir

Seyyed Ali Mohammadiyeh

Department of Pure Mathematics, Faculty of Mathematical Sciences, University of Kashan, Kashan 87317-53153 Kashan, I. R. Iran

Email: alim@kashanu.ac.ir