MATCH
Communications in Mathematical and in Computer Chemistry

MATCH Commun. Math. Comput. Chem. **82** (2019) 295-326

ISSN 0340 - 6253

Concordant Generation of Mark Tables and USCI-CF (Unit Subduced Cycle Indices with Chirality Fittingness) Tables on the Basis of Combined–Permutation Representations

Shinsaku Fujita

Shonan Institute of Chemoinformatics and Mathematical Chemistry,

Kaneko 479-7 Ooimachi, Ashigara-Kami-Gun, Kanagawa-Ken,

258-0019 Japan

shinsaku_fujita@nifty.com

(Received September 20, 2018)

Abstract

Combined-permutation representations (CPRs) have been used in the GAP (Groups, Algorithms, Programming) system to cover Fujita's USCI (Unit-Subduced-Cycle-Index) approach for symmetry-itemized enumerations of 3D structures. New GAP functions for constructing USCI-CF tables and for constructing the concordant mark tables have been developed to support the practical usage of Fujita's USCI approach. The source code containing these newely-defined functions is attached as an appendix. Concordant generation of mark tables and USCI-CF tables is applied to a CPR (degree = 4 + 2) based on a tetrahedral skeleton as well as to another CPR (degree = 10 + 2) based on an adamantane skeleton. Although these CPRs are different in their degrees, they are capable of generating an identical set of a mark table and a USCI table for the point group $\boldsymbol{T}_d$. The USCI-CF table enables us to generate a list of subduced cycle indices with chirality fittingness (SCI-CFs), which is multiplied by an inverse mark table to give a list of partial cycle indices with chirality fittingness (PCI-CFs). Each element of the list of PCI-CFs gives the PCI-CF for each subgroup. Thereby, symmetry-itemized enumeration based on a tetrahedral skeleton of $\boldsymbol{T}_d$ is conducted by means of the PCI method of Fujita's USCI approach. The results are summarized in a tabular form. The relationship between PCI-CFs and CI-CFs is discussed.

1 Introduction

Chemical compounds have long been enumerated as graphs (not 3D structures) in the conventional stereochemistry, as implied by the title of Pólya's book [1] ("Combinatorial Enumeration of Groups, Graphs, and Chemical Compounds"), which is an English translation of his famous article [2] on Pólya's theorem. After the proposal of the concept of *sphericities* [3] and of *the proligand-promolecule model* [4], the author (Fujita) has developed several approaches for the enumeration of compounds as 3D structures, the summaries of which have been published as monographs, e.g., *Fujita's proligand method* [5] for gross enumerations of 3D structures, *Fujita's USCI approach* [6,7] for symmetry-itemized enumerations, and *Fujita's stereoisogram approach* [8] for the integration of geometric features and stereoisomeric features of 3D structures.

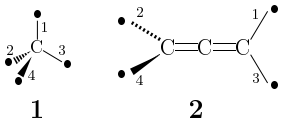
The remaining tasks are to catch up with the recent development of computer software (e.g., the GAP (Groups, Algorithms, Programming) system as a free software [9]) and to develop convenient tools for executing enumeration of compounds as 3D structures.

GAP utilities for enumeration are mainly based on permutation groups (without considering reflections), where permutations are essentially insufficient to treat 3D structures. To make up the shortcoming of permutations, the author (Fujita) has recently developed combined-permutation representations (CPRs) [10], where a permutation is combined with a mirror-permutation of 2-cycle. The CPRs have been used to treat point groups [10, 11] and *RS*-stereoisomeric groups [12]. Thereby, the GAP function `CalcConjClassCICF` for calculating cycle indices with chirality fittingness (CI-CFs) has been developed during the application of Fujita's proligand method [5] to gross enumerations of 3D structures.

The next task is to develop GAP functions for executing Fujita's USCI approach [6,7], where symmetry-itemized enumerations are based on subduced cycle indices with chirality fittingness (SCI-CFs). A brief survey of Fujita's USCI approach has appeared in Chapter 2 of Ref. [5]. Such SCI-CFs are calculated from unit subduced cycle indices with chirality fittingness (USCI-CFs), each of which is calculated by means of the subduction of a coset representation [13,14]. The present paper is devoted to develop GAP functions for calculating USCI-CFs and related matters.

2 Mark Tables (Table of Marks)

Because a tetrahedral skeleton as an *RS*-stereogenic center (so-called "chiral center") and an allene skeleton as an *RS*-stereogenic axis (so-called "chiral axis") widely attract the attention of organic chemists, the present paper deals mainly with the point group $\boldsymbol{T}_d$ of a tetrahedral skeleton **1** and the point group $\boldsymbol{D}_{2d}$ of an allene skeleton **2** (Figure 1), where the four substitution positions are numbered sequentially from 1 to 4. However, the discussions described below are applicable to any other point groups.



1 **2**

Figure 1. Reference tetrahedral skeleton **1** and allene skeleton **2**. The orbit of the four vertices of **1** corresponds to the coset representation $(\boldsymbol{C}_{3v}\backslash)\boldsymbol{T}_d$ under the point group $\boldsymbol{T}_d$. The orbit of the four vertices of **2** corresponds to the coset representation $(\boldsymbol{C}_s\backslash)\boldsymbol{D}_{2d}$ under the point group $\boldsymbol{D}_{2d}$.

The CPR of the point group $\boldsymbol{T}_d$ has been constructed during the development of computer-oriented representations for combinatorial enumerations [10, 12], where an appropriate set of generators is selected to be placed in the GAP function `Group`. The CPR named `Td_tetra` is derived from a set of generators [(1,2)(3,4), (2,3,4), (3,4)(5,6)], each of which is composed of a 4-cycle concerning the locant number 1 to 4 and a 2-cycle of mirror-permutation ((5)(6) or (5 6)). As a result, the degree of the CPR `Td_tetra` is equal to 6 (= 4 + 2). Because 1-cycles are omitted according to the notation of the GAP system, the generator (2,3,4), for example, corresponds to a full expression (1)(2 3 4)(5)(6), where commas are added for the sake of clarity. The CPR named `T_tetra` for the point group $\boldsymbol{T}$ is also constructed with omitting the 2-cycles ((5)(6) or (5 6)). The following codes are executed by inputting after the command prompt `gap>` of the GAP system.

Source-Code 1

```
gap> Td_tetra := Group([(1,2)(3,4), (2,3,4), (3,4)(5,6)]);;
gap> Size(Td_tetra); #order of Td
24
gap> T_tetra := AsSubgroup(Td_tetra,Group([(1,3)(2,4), (1,2)(3,4), (2,3,4)]));;
gap> Size(T_tetra); #order of T
12
gap> CosetDecomposition(Td_tetra,T_tetra);
[ [ (), (2,3,4), (2,4,3), (1,2)(3,4), (1,2,3), (1,2,4), (1,3,2), (1,3,4), (1,3)(2,4), (1,4,2),
    (1,4,3), (1,4)(2,3) ],
  [ (3,4)(5,6), (2,4)(5,6), (2,3)(5,6), (1,2)(5,6), (1,2,4,3)(5,6), (1,2,3,4)(5,6), (1,4,3,2)(5,6),
    (1,4)(5,6), (1,4,2,3)(5,6), (1,3,4,2)(5,6), (1,3)(5,6), (1,3,2,4)(5,6) ] ]
gap>
```

The order of the resulting $\boldsymbol{T}_d$ (`Td_tetra`) is calculated to be 24, while the order of the resulting $\boldsymbol{T}$ (`T_tetra`) is calculated to be 12. Then, the coset decomposition of $\boldsymbol{T}_d$ by its subgroup $\boldsymbol{T}$ is calculated by the GAP function `CosetDecomposition`, so as to generate two cosets, the transversals of which are I ($\sim ()$, unity) and $\sigma_{d(2)}$ ($\sim (3,4)(5,6)$, a reflection). The result shows that the second coset corresponding to $\boldsymbol{T}\sigma_{d(2)}$ ($\sigma_{d(2)} \sim (3,4)(5,6)$) is characterized by the presence of a mirror-permutation $(5,6)$.

The GAP system is originally equipped with the function `TableOfMarks` to generate a mark table (table of marks). The mark table of the group $\boldsymbol{T}_d$ (`tom_Td_tetra`) is calculated as follows:

Source-Code 2

```
gap> Td_tetra := Group([(1,2)(3,4), (2,3,4), (3,4)(5,6)]);;
gap> T_tetra := AsSubgroup(Td_tetra,Group([(1,3)(2,4), (1,2)(3,4), (2,3,4)]));;
gap> tom_Td_tetra := TableOfMarks(Td_tetra);;
gap> Display(tom_Td_tetra);
 1: 24
 2: 12 4
 3: 12 . 2
 4: 8 . . 2
 5: 6 6 . . 6
 6: 6 2 2 . . 2
 7: 6 2 . . . . 2
 8: 4 . 2 1 . . . 1
 9: 3 3 1 . 3 1 1 . 1
10: 2 2 . 2 2 . . . . 2
11: 1 1 1 1 1 1 1 1 1 1 1
```

Each row of the mark table (`tom_Td_tetra`) corresponds to a coset representation $(\boldsymbol{H}_i\backslash)\boldsymbol{T}_d$, where $\boldsymbol{H}_i$ is a subgroup up to conjugacy within $\boldsymbol{T}_d$ (`Td_tetra`). The i-th subgroup $\boldsymbol{H}_i$ (`r_tom`) can be calculated by using the GAP function `RepresentativeTom` and the corresponding set (`gen[i]`) of generators for $\boldsymbol{H}_i$ (`r_tom`) can be obtained by using the GAP function `GeneratorsOfGroup` as follows:

Source-Code 3

```
gap> gen := [];;
gap> for i in [1..Size(OrdersTom(tom_Td_tetra))] do
> r_tom := RepresentativeTom(tom_Td_tetra,i);
> gen[i] := GeneratorsOfGroup (r_tom);
> Print("gen[", i, "] := ", gen[i], "\n");
> od;
gen[1] := [ ]
gen[2] := [ (1,2)(3,4) ]
gen[3] := [ (3,4)(5,6) ]
gen[4] := [ (2,3,4) ]
gen[5] := [ (1,3)(2,4), (1,2)(3,4) ]
gen[6] := [ (3,4)(5,6), (1,2)(3,4) ]
gen[7] := [ (1,3,2,4)(5,6), (1,2)(3,4) ]
gen[8] := [ (3,4)(5,6), (2,3,4) ]
gen[9] := [ (1,3)(2,4), (1,2)(3,4), (3,4)(5,6) ]
gen[10] := [ (1,3)(2,4), (1,2)(3,4), (2,3,4) ]
gen[11] := [ (1,2)(3,4), (2,3,4), (3,4)(5,6) ]
gap> IsomorphismGroups(Group(gen[10]), T_tetra);
[ (1,3)(2,4), (1,2)(3,4), (2,3,4) ] -> [ (1,3)(2,4), (1,2)(3,4), (2,3,4) ]
```

The subgroups $\boldsymbol{H}_i$ $(i = 1, 2, \ldots, 11)$ corresponds to the subgroups of $\boldsymbol{T}_d$ as follows, where usual notations of point groups are given:

$\boldsymbol{H}_1$ (gen[1]) — $\boldsymbol{C}_1$; $\boldsymbol{H}_2$ (gen[2]) — $\boldsymbol{C}_2$; $\boldsymbol{H}_3$ (gen[3]) — $\boldsymbol{C}_s$; $\boldsymbol{H}_4$ (gen[4]) — $\boldsymbol{C}_3$;
$\boldsymbol{H}_5$ (gen[5]) — $\boldsymbol{D}_2$; $\boldsymbol{H}_6$ (gen[6]) — $\boldsymbol{C}_{2v}$; $\boldsymbol{H}_7$ (gen[7]) — $\boldsymbol{S}_4$; $\boldsymbol{H}_8$ (gen[8]) — $\boldsymbol{C}_{3v}$;
$\boldsymbol{H}_9$ (gen[9]) — $\boldsymbol{D}_{2d}$; $\boldsymbol{H}_{10}$ (gen[10]) — $\boldsymbol{T}$; and $\boldsymbol{H}_{11}$ (gen[11]) — $\boldsymbol{T}_d$;

As a result, the mark table obtained above (tom_Td_tetra) is different in the order of the appearance of subgroups from the mark table reported in a book (Table A.10 of Ref. [6]). Such mark tables of different modes should be treated flexibly because of maintaining the consistency of previous reports.

3 Concordant Generation of Mark Tables and USCI-CF Tables

To cover the results of previous reports flexibly, the order of the appearance of subgroups in a mark table is given as a list to be considered. For example, the order of subgroups collected in the mark table reported as Table A.10 of Ref. [6] is adopted as follows, where gen[5], gen[6], and gen[7] in tom_Td_tetra are sorted to give gen[7], gen[5], and gen[6]. The corresponding mark table is calculated by the newly-developed function MarkTableforUSCI (see Appendix A) as follows:

Source-Code 4

```
gap> Read("c:/fujita00/fujita2018/subductionTd/calcGAP3/USCICF.gapfunc");
gap> Td_tetra := Group([(1,2)(3,4), (2,3,4), (3,4)(5,6)]);;
gap> T_tetra := AsSubgroup(Td_tetra,Group([(1,3)(2,4), (1,2)(3,4), (2,3,4)]));;
gap> tom_Td_tetra := TableOfMarks(Td_tetra);;
gap> #Subgroups of Td given
gap> gen := [];;
gap> gen[1] := [()];; #C1
gap> gen[2] := [(1,3)(2,4)];; #C2
gap> gen[3] := [(2,4)(5,6)];; #Cs
gap> gen[4] := [(2,3,4)];; #C3
gap> gen[5] := [(1,2,3,4)(5,6)];; #S4
gap> gen[6] := [(1,3)(2,4),(1,2)(3,4)];; #D2
gap> gen[7] := [(1,3)(2,4), (2,4)(5,6)];; #C2v
gap> gen[8] := [(2,3,4), (3,4)(5,6)];; #C3v
gap> gen[9] := [(1,3)(2,4),(1,2)(3,4),(2,4)(5,6)];; #D2d
gap> gen[10] := [(1,2)(3,4), (2,3,4)];; #T
gap> gen[11] := [(1,2)(3,4), (2,3,4), (3,4)(5,6)];; #Td
gap> #mark table sorted for USCI table
gap> MarkTableTd := MarkTableforUSCI(Td_tetra,T_tetra,11,gen,4,6);;
gap> Display(MarkTableTd);
 1: 24
 2: 12 4
 3: 12 . 2
 4: 8 . . 2
 5: 6 2 . . 2
 6: 6 6 . . . 6
 7: 6 2 2 . . . 2
 8: 4 . 2 1 . . . 1
 9: 3 3 1 . 1 3 1 . 1
```

```
10: 2 2 . 2 . 2 . . . 2
11: 1 1 1 1 1 1 1 1 1 1
```

Thereby, the original mark table `tom_Td_tetra` produced above is converted into the sorted mark table `MarkTableTd`, which is equivalent to Table A.10 of Ref. [6], Table 1 of Ref. [15], and Table II of Ref. [3]. It should be noted the function `MarkTableforUSCI` is stored in the file named `USCICF.gapfunc` (Appendix A), which is loaded by means of the GAP command `Read`.

At the same time, the corresponding USCI-CF table is calculated by means of the newly-developed function `constructUSCITable` (also stored in the file `USCICF.gapfunc` (Appendix A)), as follows:

Source-Code 5

```
gap> #USCI-CF Table of Td
gap> USCITableTd := constructUSCITable(Td_tetra,T_tetra,11,gen,4,6);
[ [ b_1^24, b_2^12, c_2^12, b_3^8, c_4^6, b_4^6, c_4^6, c_6^4, c_8^3, b_12^2, c_24 ],
  [ b_1^12, b_1^4*b_2^4, c_2^6, b_3^4, c_2^2*c_4^2, b_2^6, c_2^2*c_4^2, c_6^2, c_4^3, b_6^2, c_12 ],
  [ b_1^12, b_2^6, c_2^5*a_1^2, b_3^4, c_4^3, b_4^3, c_4^2*a_2^2, c_6*a_3^2, c_8*a_4, b_12, a_12 ],
  [ b_1^8, b_2^4, c_2^4, b_1^2*b_3^2, c_4^2, b_4^2, c_4^2, c_2*c_6, c_8, b_4^2, c_8 ],
  [ b_1^6, b_1^2*b_2^2, c_2^3, b_3^2, c_4*a_1^2, b_2^3, c_2*c_4, c_6, c_4*a_2, b_6, a_6 ],
  [ b_1^6, b_1^6, c_2^3, b_3^2, c_2^3, b_1^6, c_2^3, c_6, c_2^3, b_3^2, c_6 ],
  [ b_1^6, b_1^2*b_2^2, c_2^2*a_1^2, b_3^2, c_2*c_4, b_2^3, c_4*a_1^2, a_3^2, c_4*a_2, b_6, a_6 ],
  [ b_1^4, b_2^2, c_2*a_1^2, b_1*b_3, c_4, b_4, a_2^2, a_1*a_3, a_4, b_4, a_4 ],
  [ b_1^3, b_1^3, c_2*a_1, b_3, c_2*a_1, b_1^3, c_2*a_1, a_3, c_2*a_1, b_3, a_3 ],
  [ b_1^2, b_1^2, c_2, b_1^2, c_2, b_1^2, c_2, c_2, c_2, b_1^2, c_2 ],
  [ b_1, b_1, a_1, b_1, a_1, b_1, a_1, a_1, a_1, b_1, a_1 ] ]
gap>
```

The resulting USCI-CF table (`USCITableTd`) is equivalent to Table 16 of Ref. [16] and Table E.10 of Ref. [6] (the USCI-CF a_2 at the intersection between 10th row and 5th column should be corrected to be c_2).

Such a set of codes as described above can be written in an appropriate file, which is loaded in a lump to the GAP system. For example, let a file named `D2d-USCI-CF2.gap` contain the following codes:

Source-Code 6

```
#The file D2d-USCI-CF2.gap
#Read("c:/fujita00/fujita2018/subductionTd/calcGAP3/D2d-USCI-CF2.gap");
LogTo("c:/fujita00/fujita2018/subductionTd/calcGAP3/D2d-USCI-CF2log.txt");

Read("c:/fujita00/fujita2018/subductionTd/calcGAP3/USCICF.gapfunc");

D2d := Group([(1,3)(2,4), (1,2)(3,4), (2,4)(5,6)]);
D2 := Group([(1,3)(2,4), (1,2)(3,4)]);
gen := [];
gen[1] := [ ]; #C1
gen[2] := [ (1,3)(2,4) ]; #C2
gen[3] := [ (1,2)(3,4) ]; #C2'
gen[4] := [ (2,4)(5,6) ]; #Cs
gen[5] := [ (1,3)(2,4), (1,2,3,4)(5,6) ]; #S4
gen[6] := [ (2,4)(5,6), (1,3)(5,6) ]; #C2v
gen[7] := [ (1,3)(2,4), (1,2)(3,4) ]; #D2
gen[8] := [ (1,3)(2,4), (1,2)(3,4), (2,4)(5,6) ]; #D2d
```

```
MarkTableD2d := MarkTableforUSCI(D2d,D2,8,gen,4,6);
Display("##Mark table for USCI-CF table (MarkTableD2d) : \n");
Display(MarkTableD2d);
USCITableD2d := constructUSCITable(D2d,D2,8,gen,4,6);
Display("##USCI-CF table (USCITableD2d) :");
Display(USCITableD2d);

LogTo();
```

In the above code, sets of generators for the respective subgroups up to conjugacy (gen[1]–gen[8]) are given in accord with the mark table reported previously (Table IV of Ref. [3] and Table A.8 of Ref. [6]). The concrete forms of these generators have been obtained according to the above-mentioned procedure using the GAP function `RepresentativeTom(tom_D2d,i)`, where the original mark table of $\boldsymbol{D}_{2d}$ (`tom_D2d`) is calculated by `tom_D2d := TableOfMarks(D2d)`. The resulting set of generators is sorted to meet the the mark table reported previously (gen[1]–gen[8] in Table IV of Ref. [3] and Table A.8 of Ref. [6]). Then, the mark table at issue is generated by means of the newly-developed function `MarkTableforUSCI`. The corresponding USCI-CF table is generated by means of the newly-developed function `constructUSCITable`.

For the purpose of execution, the file (`D2d-USCI-CF2.gap`) is uploaded by means of the GAP command `Read` in the command prompt. That is to say, the first sentence due to `Read` in the file is copied and pasted after the prompt word `gap>`. Thereby, the codes written in the file (Source-Code 6) are executed to give the following results in the command prompt, and simultaneously in a log file named `D2d-USCI-CF2log.txt` by means of the GAP command `LogTo`.

Source-Code 7

```
gap> Read("c:/fujita00/fujita2018/subductionTd/calcGAP3/D2d-USCI-CF2.gap");
##Mark table for USCI-CF table (MarkTableD2d) :

1: 8
2: 4 4
3: 4 . 2
4: 4 . . 2
5: 2 2 . . 2
6: 2 2 . 2 . 2
7: 2 2 2 . . . 2
8: 1 1 1 1 1 1 1 1

##USCI-CF table (USCITableD2d) :
[ [ b_1^8, b_2^4, b_2^4, c_2^4, c_4^2, c_4^2, b_4^2, c_8 ],
  [ b_1^4, b_1^4, b_2^2, c_2^2, c_2^2, c_2^2, b_2^2, c_4 ],
  [ b_1^4, b_2^2, b_1^2*b_2, c_2^2, c_4, c_4, b_2^2, c_4 ],
  [ b_1^4, b_2^2, b_2^2, c_2*a_1^2, c_4, a_2^2, b_4, a_4 ],
  [ b_1^2, b_1^2, b_2, c_2, a_1^2, c_2, b_2, a_2 ],
  [ b_1^2, b_1^2, b_2, a_1^2, c_2, a_1^2, b_2, a_2 ],
  [ b_1^2, b_1^2, b_1^2, c_2, c_2, c_2, b_1^2, c_2 ],
  [ b_1, b_1, b_1, a_1, a_1, a_1, b_1, a_1 ] ]
gap>
```

The resulting sorted mark table (`MarkTableD2d`) is equivalent to Table IV of Ref. [3] and Table A.8 of Ref. [6], which have been manually constructed by counting fixed points under the subduction of coset representations. On the other hand, the USCI-CF table (`USCITableD2d`) is equivalent to Table E.8 of Ref. [6], which has been manually constructed by examining the subduction of coset representations (Chapter 9 of Ref. [6]).

4 Generation of SCI-CFs

To pursue the process of enumeration, SCI-CFs for a given skeleton with one or more orbits should be evaluated from the respective USCI-CFs for each orbit. The four vertices of the tetrahedral skeleton **1** generates one orbit which corresponds to the coset representation $(\boldsymbol{C}_{3v}\backslash)\boldsymbol{T}_d$ under the point group $\boldsymbol{T}_d$ (Figure 1). It follows that the USCI-CFs appearing in the $(\boldsymbol{C}_{3v}\backslash)\boldsymbol{T}_d$-row (the 8th row) of the USCI-CF table (`USCITableTd`) are equivalent to SCI-CFs for the tetrahedral skeleton **1**.

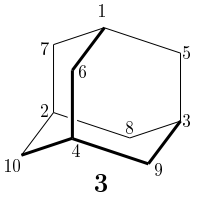
Because the CPR `Td_tetra` (degree 4 + 2) is used in the above discussion, the fixed point vector (FPV) for the coset representation $(\boldsymbol{C}_{3v}\backslash)\boldsymbol{T}_d$ appears in the 8th row of the mark table (`MarkTableTd`) or the corresponding matrix calculated by the GAP function `MatTom`. The newly-developed function `calculateFPvector` (Appendix A) is capable of calculating the fixed point vector (`FPVTd`) by starting from the CPR `Td_tetra`. Then, the newly-developed function `constructSCICF` generates the corresponding SCI-CFs, which is identical with the USCI-CFs appearing in the 8th row of the USCI-CF table (`USCITableTd`).

Source-Code 8

```
gap> #After setting data to be required
gap>
gap> Matrix_tomTd := MatTom(MarkTableTd);;
gap> Display(Matrix_tomTd);
[ [ 24, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
  [ 12, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
  [ 12, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
  [ 8, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0 ],
  [ 6, 2, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0 ],
  [ 6, 6, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0 ],
  [ 6, 2, 2, 0, 0, 0, 2, 0, 0, 0, 0, 0 ],
  [ 4, 0, 2, 1, 0, 0, 0, 1, 0, 0, 0, 0 ],
  [ 3, 3, 1, 0, 1, 3, 1, 0, 1, 0, 0, 0 ],
  [ 2, 2, 0, 2, 0, 2, 0, 0, 0, 2, 0, 0 ],
  [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 ] ]
gap>
gap> FPVTd := calculateFPvector(Td_tetra,T_tetra,11,gen,4,6);;
gap> Display(FPVTd);
[ 4, 0, 2, 1, 0, 0, 0, 1, 0, 0, 0, 0 ]
gap>
gap> l_SCICF_tetra := constructSCICF(Td_tetra,T_tetra,Matrix_tomTd,USCITableTd,FPVTd);;
gap> Display(l_SCICF_tetra);
```

```
[ b_1^4, b_2^2, c_2*a_1^2, b_1*b_3, c_4, b_4, a_2^2, a_1*a_3, a_4, b_4, a_4 ]
```

An adamantane skeleton **3** consists of two orbits, as shown in Figure 2. The four methine carbons of **3** (the locant number 1 to 4) are equivalent under the action of the point group $\boldsymbol{T}_d$. The resulting orbit corresponds to a coset representation $(\boldsymbol{C}_{3v}\backslash)\boldsymbol{T}_d$, where each position is fixed under the local symmetry $\boldsymbol{C}_{3v}$. On the other hand, the six methylene carbons (the locant number 5 to 10) construct another orbit governed by a coset representation $(\boldsymbol{C}_{2v}\backslash)\boldsymbol{T}_d$, where each position is fixed under the local symmetry $\boldsymbol{C}_{2v}$.



3

Figure 2. Adamantane skeleton having two orbits under the action of the point group $\boldsymbol{T}_d$. The one orbit of four methine carbons with the locant number 1 to 4 belongs to a coset representation $(\boldsymbol{C}_{3v}\backslash)\boldsymbol{T}_d$. The other orbit of six methylene carbons with the locant number 5 to 10 belongs to a coset representation $(\boldsymbol{C}_{2v}\backslash)\boldsymbol{T}_d$.

The FPV (`FPVadam`) for the adamantane skeleton **3** is obtained to be `[ 10, 2, 4, 1, 0, 0, 2, 1, 0, 0, 0 ]` by counting the number of fixed points under the manual application of each subgroup. Then, the GAP function `DecomposedFixedPointVector` is applied to `FPVadam`, where the data of the mark table `MarkTableTd` is used.

Source-Code 9

```
gap> #After setting data to be required
gap> FPVadam := [ 10, 2, 4, 1, 0, 0, 2, 1, 0, 0, 0 ];;
gap> DecomposedFixedPointVector(MarkTableTd,FPVadam);
[ 0, 0, 0, 0, 0, 0, 1, 1 ]
```

The resulting list indicates the participation of the 7th $(\boldsymbol{C}_{3v}\backslash)\boldsymbol{T}_d$-row and the 8th $(\boldsymbol{C}_{2v}\backslash)\boldsymbol{T}_d$-row of `MarkTableTd`. In fact, the sum of the two rows is equal to `FPVadam`.

The point group $\boldsymbol{T}_d$ can be treated as a CPR based on the adamantane skeleton **3**. The CPR named `Td_adam` is derived from a set of generators, each of which is composed of 10-cycle (4-cycle for four methine carbons and 6-cycle for methylene carbons) and a 2-cycle of mirror-permutation ((11)(12) or (11 12)). As a result, the degree of the CPR `Td_adam` is equal to 12 (= 10 + 2). The CPR named `T_adam` for the point group $\boldsymbol{T}$ is

also constructed with omitting the 2-cycles ((11)(12) or (11 12)).

Source-Code 10 (stored as `Td-adamX.gap`)

```
#Read("c:/fujita00/fujita2018/subductionTd/calcGAP3/Td-adamX.gap");
LogTo("c:/fujita00/fujita2018/subductionTd/calcGAP3/Td-adamXlog.txt");

Read("c:/fujita00/fujita2018/subductionTd/calcGAP3/USCICF.gapfunc");

Td_adam := Group([(1,2)(3,4)(5,10)(6,8), (2,3,4)(5,6,7)(8,9,10), (3,4)(5,6)(8,10)(11,12)]);;
T_adam := AsSubgroup(Td_adam,
          Group([(1,3)(2,4)(7,9)(6,8),(1,2)(3,4)(5,10)(6,8), (2,3,4)(5,6,7)(8,9,10)]));;

tom_Td_adam := TableOfMarks(Td_adam);;
#Display(tom_Td_adam); #identical with tom_Td_tetra

##Subgroups of Td given
gen := [ ];;
gen[1] := [ ]; #C1
gen[2] := [ ( 1, 2)( 3, 4)( 5,10)( 6, 8) ]; #C2
gen[3] := [ ( 3, 4)( 5, 6)( 8,10)(11,12) ]; #Cs
gen[4] := [ ( 2, 3, 4)( 5, 6, 7)( 8, 9,10) ]; #C3
gen[6] := [ ( 1, 3)( 2, 4)( 6, 8)( 7, 9), ( 1, 2)( 3, 4)( 5,10)( 6, 8) ]; #D2
gen[7] := [ ( 3, 4)( 5, 6)( 8,10)(11,12), ( 1, 2)( 3, 4)( 5,10)( 6, 8) ]; #C2v
gen[5] := [ ( 1, 3, 2, 4)( 5, 8,10, 6)( 7, 9)(11,12), ( 1, 2)( 3, 4)( 5,10)( 6, 8) ]; #S4
gen[8] := [ ( 3, 4)( 5, 6)( 8,10)(11,12), ( 2, 3, 4)( 5, 6, 7)( 8, 9,10) ]; #C3v
gen[9] := [ ( 1, 3)( 2, 4)( 6, 8)( 7, 9), ( 1, 2)( 3, 4)( 5,10)( 6, 8), ( 3, 4)( 5, 6)( 8,10)(11,12) ];
        #D2d
gen[10] := [ ( 1, 3)( 2, 4)( 6, 8)( 7, 9), ( 1, 2)( 3, 4)( 5,10)( 6, 8), ( 2, 3, 4)( 5, 6, 7)( 8, 9,10) ];
        #T
gen[11] := [ ( 1, 2)( 3, 4)( 5,10)( 6, 8), ( 2, 3, 4)( 5, 6, 7)( 8, 9,10), ( 3, 4)( 5, 6)( 8,10)(11,12) ];
        #Td

#USCI Table of Td
USCITableTdadam := constructUSCITable(Td_adam,T_adam,11,gen,10,12);;
#Display(USCITableTdadam); #identical with USCITableTd

#mark table sorted for USCI table
MarkTableTdadam := MarkTableforUSCI(Td_adam,T_adam,11,gen,10,12);;
#Display(MarkTableTdadam); #identical with MarkTableTd

#Matrix form of mark table
Matrix_tomTdadam := MatTom(MarkTableTdadam);
#Display(Matrix_tomTdadam); #identical with Matrix_tomTd

Display("#Fixed point vector for adamantane");
FPVadam := calculateFPvector(Td_adam,T_adam,11,gen,10,12);;
Display(FPVadam);

Display("#SCI-CF for adamantane");
l_SCICF_adam := constructSCICF(Td_adam,T_adam,Matrix_tomTdadam,USCITableTdadam,FPVadam);;
Display(l_SCICF_adam);

LogTo();
```

The mark table `tom_Td_adam` calculated by `TableOfMarks(Td_adam)` (Source-Code 10) is identical with `tom_Td_tetra` described above (Source-Code 2). The list of subgroups of $\boldsymbol{T}_d$ is given by sorting the results obtained in accord with the CPR of degree 12. Note that the order of appearance is maintained due to the output of `tom_Td_adam`, while the sequence number i in the gen[i] is renumbered to match the sorted mark table `MarkTableTdadam` (Source-Code 10). The sorted mark table `MarkTableTdadam` calculated by using the function `MarkTableforUSCI` is identical with `MarkTableTd` described above (Source-Code 4). The corresponding USCI-CF table (`USCITableTdadam`) is identical with

`USCITableTd` described above (Source-Code 5).

The source-code file `TdadamX.gap` (Source-Code 10) is loaded by the GAP function `Read`. Thereby, the FPV (`FPVadam` due to `calculateFPvector`) for the adamantane skeleton **3** and the set of SCI-CFs (`l_SCICF_adam` due to `constructSCICF`) are calculated as follows.

Source-Code 11

```
gap> Read("c:/fujita00/fujita2018/subductionTd/calcGAP3/Td-adamX.gap");
#Fixed point vector for adamantane
[ 10, 2, 4, 1, 0, 0, 2, 1, 0, 0, 0 ]
#SCI-CF for adamantane
[ b_1^10, b_1^2*b_2^4, c_2^3*a_1^4, b_1*b_3^3, c_2*c_4^2, b_2^3*b_4, c_4*a_1^2*a_2^2, a_1*a_3^3,
  c_4*a_2*a_4, b_4*b_6, a_4*a_6 ]
```

The SCI-CFs can alternatively be derived by referring to the 7th ($\boldsymbol{C}_{3v}\backslash\rangle\boldsymbol{T}_d$-row and the 8th ($\boldsymbol{C}_{2v}\backslash\rangle\boldsymbol{T}_d$-row of the USCI-CF table `USCITableTd` (Source-Code 5) or of the USCI-CF table `USCITableTdadam` (calculated in Source-Code 10, but not printed).

5 Generation of PCI-CFs

Fujita's USCI approach provides us with four methods of symmetry-itemized enumeration [6, 7], i.e., the fixed-point matrix (FPM) method [13, 17, 18], the partial-cycle-index (PCI) method [19, 20], the elementary-superposition (ES) method [21], and the partial-superposition (PS) method [19, 21]. Among them, the FPM method [22], the PCI method [23], and the ES method [24] have been applied to the symmetry-itemized enumeration of cubane derivatives as common targets.

Because the GAP system has convenient functions of treating polynomials for group theory, the PCI-CF method (cf. Sections 16.3 and 19.5 of Ref. [6]) is one of the best choices for developing practical devices for the symmetry-itemized enumeration, where partial cycle indices (PCI-CFs) for respective subgroups are calculated as polynomials from a list of SCI-CFs (such as `l_SCICF_tetra` and `l_SCICF_adam`).

Let us exemplify the procedure of calculating a list of PCI-CFs for a tetrahedral skeleton **1**. Definition 19.6 of Ref. [6] teaches us that a list of PCI-CFs (`l_PCICF_tetra`) is obtained by the multiplication of a list of SCI-CFs and the inverse mark table, i.e., `l_SCICF_tetra * invMatrix_tomTd`, where the inverse mark table (`invMatrix_tomTd`) is calculated by the GAP function (`Inverse`) from the mark table (`Matrix_tomTd`) obtained above (Source-Code 8). The following source list is executed by the GAP system.

Source-Code 12

```
gap> #Matrix_tomTd calculated above (Source-Code 8)
gap> Matrix_tomTd :=
> [ [ 24, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],

  (omitted )

> [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 ] ];;
gap> #Inverse mark table of Matrix_tomTd
gap> invMatrix_tomTd := Inverse(Matrix_tomTd);
[ [ 1/24, 0, 0, 0, 0, 0, 0, 0, 0, 0 ], [ -1/8, 1/4, 0, 0, 0, 0, 0, 0, 0, 0 ],
  [ -1/4, 0, 1/2, 0, 0, 0, 0, 0, 0, 0 ], [ -1/6, 0, 0, 1/2, 0, 0, 0, 0, 0, 0 ],
  [ 0, -1/4, 0, 0, 1/2, 0, 0, 0, 0, 0 ], [ 1/12, -1/4, 0, 0, 0, 1/6, 0, 0, 0, 0 ],
  [ 1/4, -1/4, -1/2, 0, 0, 0, 1/2, 0, 0, 0 ], [ 1/2, 0, -1, -1/2, 0, 0, 0, 1, 0, 0 ],
  [ 0, 1/2, 0, 0, -1/2, -1/2, -1/2, 0, 1, 0 ], [ 1/6, 0, 0, -1/2, 0, -1/6, 0, 0, 0, 1/2, 0 ],
  [ -1/2, 0, 1, 1/2, 0, 1/2, 0, -1, -1, -1/2, 1 ] ]
gap> #Definition of variables
gap> b_1 := Indeterminate(Rationals, "b_1");; b_2 := Indeterminate(Rationals, "b_2");;
gap> b_3 := Indeterminate(Rationals, "b_3");; b_4 := Indeterminate(Rationals, "b_4");;
gap> a_1 := Indeterminate(Rationals, "a_1");; a_2 := Indeterminate(Rationals, "a_2");;
gap> a_3 := Indeterminate(Rationals, "a_3");; a_4 := Indeterminate(Rationals, "a_4");;
gap> c_2 := Indeterminate(Rationals, "c_2");; c_4 := Indeterminate(Rationals, "c_4");;
gap> #List of SCI-CF calculated above (Source-Code 8)
gap> l_SCICF_tetra :=
> [ b_1^4, b_2^2, c_2*a_1^2, b_1*b_3, c_4, b_4, a_2^2, a_1*a_3, a_4, b_4, a_4 ];;
gap> #List of PCI-CF calculated from list of SCI-CF
gap> l_PCICF_tetra := l_SCICF_tetra * invMatrix_tomTd;
[ 1/24*b_1^4-1/4*a_1^2*c_2-1/6*b_1*b_3-1/8*b_2^2+1/2*a_1*a_3+1/4*a_2^2+1/4*b_4-1/2*a_4,
  1/4*b_2^2-1/4*a_2^2-1/4*b_4+1/2*a_4-1/4*c_4, 1/2*a_1^2*c_2-a_1*a_3-1/2*a_2^2+a_4,
  1/2*b_1*b_3-1/2*a_1*a_3-1/2*b_4+1/2*a_4, -1/2*a_4+1/2*c_4, 0, 1/2*a_2^2-1/2*a_4, a_1*a_3-a_4,
  0, 1/2*b_4-1/2*a_4,a_4 ]
gap> #PCI-CFs for subgroups
gap> for i in [1..11] do
> Print("PCI-CF[", i, "] := ", l_PCICF_tetra[i], "\n");
> od;
PCI-CF[1] := 1/24*b_1^4-1/4*a_1^2*c_2-1/6*b_1*b_3-1/8*b_2^2+1/2*a_1*a_3+1/4*a_2^2+1/4*b_4-1/2*a_4
PCI-CF[2] := 1/4*b_2^2-1/4*a_2^2-1/4*b_4+1/2*a_4-1/4*c_4
PCI-CF[3] := 1/2*a_1^2*c_2-a_1*a_3-1/2*a_2^2+a_4
PCI-CF[4] := 1/2*b_1*b_3-1/2*a_1*a_3-1/2*b_4+1/2*a_4
PCI-CF[5] := -1/2*a_4+1/2*c_4
PCI-CF[6] := 0
PCI-CF[7] := 1/2*a_2^2-1/2*a_4
PCI-CF[8] := a_1*a_3-a_4
PCI-CF[9] := 0
PCI-CF[10] := 1/2*b_4-1/2*a_4
PCI-CF[11] := a_4
gap> #CI-CF derived from PCI-CFs
gap> sum_l_PCICF_tetra := Sum(l_PCICF_tetra);;
gap> Print("CICF_Td := ", sum_l_PCICF_tetra, "\n");
CICF_Td := 1/24*b_1^4+1/4*a_1^2*c_2+1/3*b_1*b_3+1/8*b_2^2+1/4*c_4
gap>
```

Each element of the list of PCI-CFs (`l_PCICF_tetra`) is the PCI-CF (PCI-CF[i]) corresponding to the respective subgroup numbered sequentially, where the sets of generators: gen[i] ($i = 1, 2, \ldots, 11$) correspond to `PCI-CF`[i] for the point groups $\boldsymbol{C}_1$, $\boldsymbol{C}_2$, $\boldsymbol{C}_s$, $\boldsymbol{C}_3$, $\boldsymbol{S}_4$, $\boldsymbol{D}_2$, $\boldsymbol{C}_{2v}$, $\boldsymbol{C}_{3v}$, $\boldsymbol{D}_{2d}$, $\boldsymbol{T}$, and $\boldsymbol{T}_d$, respectively. Hence, the above PCI-CFs (PCI-CF[i]) can be written in usual notation (cf. Definition 19.6 of Ref. [6]) as follows:

$$\text{PCI-CF}(\boldsymbol{C}_1, \$_d) = \frac{1}{24}b_1^4 - \frac{1}{4}a_1^2c_2 - \frac{1}{6}b_1b_3 - \frac{1}{8}b_2^2 + \frac{1}{2}a_1a_3 + \frac{1}{4}a_2^2 + \frac{1}{4}b_4 - \frac{1}{2}a_4 \tag{1}$$

$$\text{PCI-CF}(\boldsymbol{C}_2, \$_d) = \frac{1}{4}b_2^2 - \frac{1}{4}a_2^2 - \frac{1}{4}b_4 + \frac{1}{2}a_4 - \frac{1}{4}c_4 \tag{2}$$

$$\text{PCI-CF}(\boldsymbol{C}_s, \$_d) = \frac{1}{2}a_1^2 c_2 - a_1 a_3 - \frac{1}{2}a_2^2 + a_4 \tag{3}$$

$$\text{PCI-CF}(\boldsymbol{C}_3, \$_d) = \frac{1}{2}b_1 b_3 - \frac{1}{2}a_1 a_3 - \frac{1}{2}b_4 + \frac{1}{2}a_4 \tag{4}$$

$$\text{PCI-CF}(\boldsymbol{S}_4, \$_d) = -\frac{1}{2}a_4 + \frac{1}{2}c_4 \tag{5}$$

$$\text{PCI-CF}(\boldsymbol{D}_2, \$_d) = 0 \tag{6}$$

$$\text{PCI-CF}(\boldsymbol{C}_{2v}, \$_d) = \frac{1}{2}a_2^2 - \frac{1}{2}a_4 \tag{7}$$

$$\text{PCI-CF}(\boldsymbol{C}_{3v}, \$_d) = a_1 a_3 - a_4 \tag{8}$$

$$\text{PCI-CF}(\boldsymbol{D}_{2d}, \$_d) = 0 \tag{9}$$

$$\text{PCI-CF}(\boldsymbol{T}, \$_d) = \frac{1}{2}b_4 - \frac{1}{2}a_4 \tag{10}$$

$$\text{PCI-CF}(\boldsymbol{T}_d, \$_d) = a_4 \tag{11}$$

where the symbol $\$_d$ represents a_d, c_d, or b_d and the subscript d represents a positive integer.

6 Enumeration by the PCI-CF method

Suppose that the four positions of the tetrahedral skeleton **1** are occupied by a set of four proligands selected from the following ligand inventory:

$$\boldsymbol{L} = \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{p}/\overline{\mathrm{p}}, \mathrm{q}/\overline{\mathrm{q}}, \mathrm{r}/\overline{\mathrm{r}}, \mathrm{s}/\overline{\mathrm{s}}, \}, \tag{12}$$

where the uppercase letters, A, B, C, and D, indicate achiral proligands, while a pair of lowercase letters without and with an overline, p/$\overline{\mathrm{p}}$, q/$\overline{\mathrm{q}}$, r/$\overline{\mathrm{r}}$, or s/$\overline{\mathrm{s}}$, indicates a pair of enantiomeric proligands when detached. Then, the following inventory-functions are calculated:

$$a_d = \mathrm{A}^d + \mathrm{B}^d + \mathrm{C}^d + \mathrm{D}^d \quad \text{(no lowercase terms)} \tag{13}$$

$$c_d = \mathrm{A}^d + \mathrm{B}^d + \mathrm{C}^d + \mathrm{D}^d + 2\mathrm{p}^{d/2}\overline{\mathrm{p}}^{d/2} + 2\mathrm{q}^{d/2}\overline{\mathrm{q}}^{d/2} + 2\mathrm{r}^{d/2}\overline{\mathrm{r}}^{d/2} + 2\mathrm{s}^{d/2}\overline{\mathrm{s}}^{d/2} \tag{14}$$

$$b_d = \mathrm{A}^d + \mathrm{B}^d + \mathrm{C}^d + \mathrm{D}^d + \mathrm{p}^d + \overline{\mathrm{p}}^d + \mathrm{q}^d + \overline{\mathrm{q}}^d + \mathrm{r}^d + \overline{\mathrm{r}}^d + \mathrm{s}^d + \overline{\mathrm{s}}^d \tag{15}$$

These inventory-functions are introduced into the right-hand side of each PCI-CF (Eqs. 1–11). The resulting equation is expanded to give a generating function, in which the coefficient of the term $\mathrm{A}^a\mathrm{B}^b\mathrm{C}^c\mathrm{D}^d\mathrm{p}^p\overline{\mathrm{p}}^{\overline{p}}\mathrm{q}^q\overline{\mathrm{q}}^{\overline{q}}\mathrm{r}^r\overline{\mathrm{r}}^{\overline{r}}\mathrm{s}^s\overline{\mathrm{s}}^{\overline{s}}$ (representing the composition at issue) indicates the number of isomeric promolecules belonging to the corresponding subgroup. The composition is represented by the following partition:

$$[\theta] = [a, b, c, d; p, \overline{p}, q, \overline{q}, r, \overline{r}, s, \overline{s}], \tag{16}$$

where the respective elements represent non-negative integers which satisfy the following condition:

$$a+b+c+d+p+\overline{p}+q+\overline{q}+r+\overline{r}+s+\overline{s}=4. \tag{17}$$

Because of symmetric appearance of the terms, the restriction condition, $a \geq b \geq c \geq d$ as well as $p \geq q \geq r \geq s$ ($p \geq \overline{p}$, $q \geq \overline{q}$, $r \geq \overline{r}$, $s \geq \overline{s}$), is postulated without losing generality.

The procedures of generating the generating functions from the PCI-CFs and of calculating the coefficients of respective compositions (in the file `enum-tetra.gap` attached as Appendix B) are similar to the procedures concerning CI-CFs [11]. Note that the GAP functions developed to treat CI-CFs [11], e.g., `calcCoeffGen`, are capable of treat PCI-CFs as they are. Hence, the file `CICFgenCC.gapfunc` (Appendix A of [11]) should be loaded to use the function `calcCoeffGen` along a similar way to Appendix B of [11].

The results obtained from the attached Appendix B are summarized in Table 1. The value in a row attached by an asterisk (*) should be duplicated because the value 1/2 means the presence of a pair of enantiomers. For example, the partition $[3,0,0,0,1,0,\ldots,0]$ ($\mathrm{A}^3\mathrm{p}$) is coupled with the counterpart partition $[3,0,0,0,0,1,\ldots,0]$ ($\mathrm{A}^3\overline{\mathrm{p}}$) in the form of $\frac{1}{2}(\mathrm{A}^3\mathrm{p}+\mathrm{A}^3\overline{\mathrm{p}})$, which indicates a pair of enantiomers.

In contrast, the value in the row without an asterisk should be used as it is, so that it may indicate the number of achiral or chiral promolecules. For exmaple, the value 2 at the intersection between the $[1,1,0,0,1,1,0,0,0,0,0,0]$-row and the $\boldsymbol{C}_s$-column indicates the presence of two achiral promolecules, e.g., two *RS*-diastereomers having the composition $\mathrm{ABp}\overline{\mathrm{p}}$ [4]. The two *RS*-diastereomers of $\mathrm{ABp}\overline{\mathrm{p}}$ are paired to give a pair of *RS*-diastereomers, which is characterized by a type-V stereoisogram [25]. On the other hand, the value 1 at the intersection between the $[0,0,0,0,1,1,1,1,0,0,0,0]$-row and the $\boldsymbol{C}_1$-column indicates the presence of one pair of enantiomeric promolecules [4], e.g., a pair of enantiomers with the composition $\mathrm{p}\overline{\mathrm{p}}\mathrm{q}\overline{\mathrm{q}}$, which is characterized by a type-I stereoisogram [25].

The values of Table 1 are consistent with the data reported previously in Table 1 of Ref. [4] and Table 21.1 of Ref. [6], which are obtained in terms of the fixed-point matrix (FPM) Method of Fujita's USCI approach [22].

Table 1. The Symmetry-Itemized Enumeration of Isomers Derived from a Tetrahedral Skeleton on the Basis of the PCI-CF Method

$[\theta]$	$\boldsymbol{C}_1$	$\boldsymbol{C}_2$	$\boldsymbol{C}_s$	$\boldsymbol{C}_3$	$\boldsymbol{S}_4$	$\boldsymbol{D}_2$	$\boldsymbol{C}_{2v}$	$\boldsymbol{C}_{3v}$	$\boldsymbol{D}_{2d}$	$\boldsymbol{T}$	$\boldsymbol{T}_d$
$[4,0,0,0,0,0,0,0,0,0,0,0]$	0	0	0	0	0	0	0	0	0	0	1
$[3,1,0,0,0,0,0,0,0,0,0,0]$	0	0	0	0	0	0	0	1	0	0	0
$[3,0,0,0,1,0,0,0,0,0,0,0]$*	0	0	0	1/2	0	0	0	0	0	0	0
$[2,2,0,0,0,0,0,0,0,0,0,0]$	0	0	0	0	0	0	1	0	0	0	0
$[2,0,0,0,2,0,0,0,0,0,0,0]$*	0	1/2	0	0	0	0	0	0	0	0	0
$[2,1,1,0,0,0,0,0,0,0,0,0]$	0	0	1	0	0	0	0	0	0	0	0
$[2,1,0,0,1,0,0,0,0,0,0,0]$*	1/2	0	0	0	0	0	0	0	0	0	0
$[2,0,0,0,1,1,0,0,0,0,0,0]$	0	0	1	0	0	0	0	0	0	0	0
$[2,0,0,0,1,0,1,0,0,0,0,0]$*	1/2	0	0	0	0	0	0	0	0	0	0
$[1,1,1,1,0,0,0,0,0,0,0,0]$	1	0	0	0	0	0	0	0	0	0	0
$[1,1,1,0,1,0,0,0,0,0,0,0]$*	1	0	0	0	0	0	0	0	0	0	0
$[1,1,0,0,2,0,0,0,0,0,0,0]$*	1/2	0	0	0	0	0	0	0	0	0	0
$[1,1,0,0,1,1,0,0,0,0,0,0]$	0	0	2	0	0	0	0	0	0	0	0
$[1,1,0,0,1,0,1,0,0,0,0,0]$*	1	0	0	0	0	0	0	0	0	0	0
$[1,0,0,0,3,0,0,0,0,0,0,0]$*	0	0	0	1/2	0	0	0	0	0	0	0
$[1,0,0,0,2,1,0,0,0,0,0,0]$*	1/2	0	0	0	0	0	0	0	0	0	0
$[1,0,0,0,2,0,1,0,0,0,0,0]$*	1/2	0	0	0	0	0	0	0	0	0	0
$[1,0,0,0,1,1,1,0,0,0,0,0]$*	1	0	0	0	0	0	0	0	0	0	0
$[1,0,0,0,1,0,1,0,1,0,0,0]$*	1	0	0	0	0	0	0	0	0	0	0
$[0,0,0,0,4,0,0,0,0,0,0,0]$*	0	0	0	0	0	0	0	0	0	1/2	0
$[0,0,0,0,3,1,0,0,0,0,0,0]$*	0	0	0	1/2	0	0	0	0	0	0	0
$[0,0,0,0,3,0,1,0,0,0,0,0]$*	0	0	0	1/2	0	0	0	0	0	0	0
$[0,0,0,0,2,2,0,0,0,0,0,0]$	0	0	0	0	1	0	0	0	0	0	0
$[0,0,0,0,2,1,1,0,0,0,0,0]$*	1/2	0	0	0	0	0	0	0	0	0	0
$[0,0,0,0,2,0,2,0,0,0,0,0]$*	0	1/2	0	0	0	0	0	0	0	0	0
$[0,0,0,0,2,0,1,1,0,0,0,0]$*	1/2	0	0	0	0	0	0	0	0	0	0
$[0,0,0,0,2,0,1,0,1,0,0,0]$*	1/2	0	0	0	0	0	0	0	0	0	0
$[0,0,0,0,1,1,1,1,0,0,0,0]$	1	0	0	0	0	0	0	0	0	0	0
$[0,0,0,0,1,1,1,0,1,0,0,0]$*	1	0	0	0	0	0	0	0	0	0	0
$[0,0,0,0,1,0,1,0,1,0,1,0]$*	1	0	0	0	0	0	0	0	0	0	0

* To be duplicated.

7 PCI-CFs vs. CI-CFs

The summation of PCI-CFs for symmetry-itemized enumeration generates a cycle index with chirality fittingness (CI-CF) for gross enumeration (Definition 16.5, 19.7 and 19.8 of Ref. [6]). The last part of Source-Code 12 indicates that the application of the GAP function `Sum` to the list of PCI-CFs for the tetrahedral skeleton **1** (`l_PCICF_tetra`) give the sum of Eqs. 1–11 (`sum_l_PCICF_tetra`), which is equal to the CI-CF for **1**. The summation procedure can be written in usual notation as follows:

$$\begin{aligned}\text{CI-CF}(\boldsymbol{T}_d,\$) =&\text{PCI-CF}(\boldsymbol{C}_1,\$) + \text{PCI-CF}(\boldsymbol{C}_2,\$) + \text{PCI-CF}(\boldsymbol{C}_s,\$) + \text{PCI-CF}(\boldsymbol{C}_3,\$)\\ &+ \text{PCI-CF}(\boldsymbol{S}_4,\$) + \text{PCI-CF}(\boldsymbol{D}_2,\$) + \text{PCI-CF}(\boldsymbol{C}_{2v},\$)\\ &+ \text{PCI-CF}(\boldsymbol{C}_{3v},\$) + \text{PCI-CF}(\boldsymbol{D}_{2d},\$) + \text{PCI-CF}(\boldsymbol{T},\$) + \text{PCI-CF}(\boldsymbol{T}_d,\$)\\ =&\frac{1}{24}b_1^4 + \frac{1}{4}a_1^2c_2 + \frac{1}{3}b_1b_3 + \frac{1}{8}b_2^2 + \frac{1}{4}c_4 \end{aligned} \tag{18}$$

The coefficients appearing in the CI-CF (e.g., Eq. 18) are obtained from the corresponding inverse mark table in accord with Theorem 16.2 of Ref. [6] and Theorem 2.8 of Ref. [5]. A practical calculation due to the GAP system is shown in Source-Code 13.

Source-Code 13

```
gap> #Inverse mark table of Matrix_tomTd calculated above (Source-Code 12)
gap> invMatrix_tomTd :=
> [ [ 1/24, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ], [ -1/8, 1/4, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],

    (omitted)

> [ -1/2, 0, 1, 1/2, 0, 1/2, 0, -1, -1, -1/2, 1 ] ];;
gap> #sum of each row in inverse mark table (invMatrix_tomTd)
gap> row_sum := [];;
gap> for i in [1..11] do
> row_sum[i] := Sum(invMatrix_tomTd[i]);
> od;
gap> Display(row_sum);
[ 1/24, 1/8, 1/4, 1/3, 1/4, 0, 0, 0, 0, 0, 0 ]
gap> #Definition of variables
gap> b_1 := Indeterminate(Rationals, "b_1");; b_2 := Indeterminate(Rationals, "b_2");;
gap> b_3 := Indeterminate(Rationals, "b_3");; b_4 := Indeterminate(Rationals, "b_4");;
gap> a_1 := Indeterminate(Rationals, "a_1");; a_2 := Indeterminate(Rationals, "a_2");;
gap> a_3 := Indeterminate(Rationals, "a_3");; a_4 := Indeterminate(Rationals, "a_4");;
gap> c_2 := Indeterminate(Rationals, "c_2");; c_4 := Indeterminate(Rationals, "c_4");;
gap> #List of SCI-CF calculated above (Source-Code 8)
gap> l_SCICF_tetra :=
> [ b_1^4, b_2^2, c_2*a_1^2, b_1*b_3, c_4, b_4, a_2^2, a_1*a_3, a_4, b_4, a_4 ];;
gap> #CI-CF derived from SCI-CFs
gap> ip_l_SCICF_tetra := l_SCICF_tetra*row_sum;;
gap> Print("CICF_Td_x := ", ip_l_SCICF_tetra, "\n");
CICF_Td_x := 1/24*b_1^4+1/4*a_1^2*c_2+1/3*b_1*b_3+1/8*b_2^2+1/4*c_4
gap>
```

The properties of inverse mark tables such as `invMatrix_tomTd` have been generally discussed in Chapter 2 and Appendix B of Ref. [5].

As shown in Source-Code 13, the application of the GAP function `Sum` to each row of the inverse mark table (`invMatrix_tomTd`) gives a column vector, which is then transposed into a list of sums (`row_sum`) as a row vector. The resulting list (`row_sum`) [1/24, 1/8, 1/4, 1/3, 1/4, 0, ...] contains positive fractional values for respective cyclic subgroups and zero values for respective non-cyclic subgroups. This is consistent with the general theorem previously reported (Theorem 16.2 of Ref. [6] and Theorem 2.8 of Ref. [5]).

The list of SCI-CFs calculated in Source-Code 8 (`l_SCICF_tetra`) is multiplied by the list of coefficients (`row_sum`) to give the corresponding inner product (`ip_l_SCICF_tetra`), which is identical with the CI-CF (`sum_l_PCICF_tetra` of Source-Code 12, Eq. 18).

On the other hand, the function `CalcConjClassCICF` was developed to calculate CI-CFs according to Fujita's proligand method [5]. The source code was delivered as a file named `CICFgenCC.gapfunc` (Appendix A of Ref. [11]). The following Source-Code 14 indicates that the function `CalcConjClassCICF` is capable of calculating the CI-CF for the tetrahedral skeleton **1**. The resulting CI-CF (`CICF_Td`) is identical with the above-mentioned inner product (`ip_l_SCICF_tetra` of Source-Code 13) as well as the CI-CF (`sum_l_PCICF_tetra` of Source-Code 12, Eq. 18).

Source-Code 14

```
gap> #Loading of CICFgenCC.gapfunction
gap> Read("c:/fujita00/fujita2018/subductionTd/calcGAP3/CICFgenCC.gapfunc");
gap> #Fujita's CI-CF
gap> Td_tetra := Group([(1,2)(3,4), (2,3,4), (3,4)(5,6)]);;
gap> CICF_Td := CalcConjClassCICF(Td_tetra, 4, 6);
1/24*b_1^4+1/4*a_1^2*c_2+1/3*b_1*b_3+1/8*b_2^2+1/4*c_4
gap>
```

It should be noted that the CI-CFs due to Fujita's proligand method (e.g., Source-Code 14) are based on the concept of the sphericities of cycles [26], while the counterparts due to Fujita's USCI approach (e.g., Source-Codes 12 and 13) are based on the concept of the sphericity of orbits for cyclic subgroups [26] (cf. Section 7.2 of Ref. [5]).

8 Conclusion

Combined-permutation representations (CPRs), which were originally developed for the application of Fujita's proligand method [5] to gross enumerations of 3D structures, have been used to cover Fujita's USCI approach [6] for symmetry-itemized enumerations of 3D structures. New GAP functions for constructing USCI-CF tables (`constructUSCITable`) and for constructing the concordant mark tables (`MarkTableforUSCI`) have been devel-

oped to support the practical usage of Fujita's USCI approach (Appendix A). Concordant generation of mark tables and USCI-CF tables is applied to a CPR (degree = 4 + 2) based on a tetrahedral skeleton (Source-Codes 4 and 5) and to another CPR (degree = 10 + 2) based on an adamantane skeleton (Source-Code 10), so that these CPRs are capable of generating an identical set of a mark table and a USCI-Table for the point group $\boldsymbol{T}_d$. One or more rows of a USCI-CF table generate(s) a list of subduced cycle indices with chirality fittingness (SCI-CFs), which is multiplied by an inverse mark table to give a list of partial cycle indices with chirality fittingness (PCI-CFs). Each element of the list of PCI-CFs gives the PCI-CF for each subgroup (Source-Code 12). Thereby, symmetry-itemized enumeration based on a tetrahedral skeleton of $\boldsymbol{T}_d$ is conducted by means of the PCI method of Fujita's USCI approach (Appendix B). The results are summarized in Table 1. The relationship between PCI-CFs and CI-CFs is discussed.

Appendix A. `USCICF.gapfunc` Containing Functions for Concordant Generation of Mark Tables and USCI-CF Tables

The file `USCICF.gapfunc` contains several basic functions, i.e., `CosetRepCF` for calculating a coset representation (CR); `detectTomSubgroup` for detecting a row corresponding to the local subgroup of the CR; `constructUSCICF` for construction of a USCI-CF; `constructUSCICFlist` for construction of the list of USCI-CFs corresponding to the CR; and `constructSCICFlist` for construction of the list of USCI-CFs corresponding to the CR. Thereafter, the file `USCICF.gapfunc` defines several utility functions, i.e., `constructUSCITable` for construction of a USCI-CF table (for examples, see Source-Code 5, Source-Code 6, and Source-Code 10); `MarkTableforUSCI` for construction of a mark table corresponding to a USCI-CF table due to function `constructUSCITable` (for examples, see Source-Code 4, Source-Code 6, and Source-Code 10); `constructSCICF` for construction of the list of SCI-CFs (or USCI-CFs) corresponding to a set of CRs (for examples, see Source-Code 8 and Source-Code 10); and `calculateFPvector` for calculating a fixed-point vector (FPV) which corresponds to a group derived by a given set of generators (for examples, see Source-Code 8 and Source-Code 10).

(`USCICF.gapfunc`)

```
#Read("c:/fujita00/fujita2018/subductionTd/calcGAP3/USCICF.gapfunc");
```

```
#MakeReadWriteGlobal("Glgrp");; #global symmetry
Glgrp := Group([()]);; #tentative setting for global symmetry
#MakeReadWriteGlobal("Locgrp");; #local symmetry
Locgrp := Group([()]);; #tentative setting for local symmetry
#MakeReadWriteGlobal("MxChgrp");; #maximum chiral subgroup
MxChgrp := Group([()]);; #tentative setting for max chiral subgroup
#MakeReadWriteGlobal("row_list");; #fixed-point vector
row_list := [];; #tentative setting for fixed-point vector
#MakeReadWriteGlobal("DegCGr");; #degree
DegCGr := 0;; #tentative setting for degree
#MakeReadWriteGlobal("DegGr");; #full degree
DegGr := 0;; #tentative setting for full degree
#MakeReadWriteGlobal("tom_Glgrp");; #table of marks of Glgrp
tom_Glgrp := TableOfMarks(Glgrp);; #tentative setting for tom of Glgrp
#####################################################
#####################################################
## Function for Calculating a Coset Representation #
## globalgr(/localgrp) #
#####################################################
fixedpoint := 1; #Global fixed point (default)
isstabilizer := 1; #Global stabilizer or not
CosetRepCF := function(globalgrp,localgrp,maxchgrp,degree,degreefull)
#CosetRepCF := function(Glgrp,Locgrp,MxChgrp,DegCGr,DegGr)
local i, j, k,
#Glgrp, Locgrp, MxChgrp, DegCGr, DegGr,
l_elm_Glgrp, l_elm_MxChgrp,
cd_Gl_MxC,cd_Gl_Loc, l_rep, calcdegree,
perm_cd, s_perm_cd, l_perm, ll_perm, cosetrep;
Glgrp := globalgrp;
Locgrp := localgrp;
MxChgrp := maxchgrp;
DegCGr := degree; DegGr := degreefull;
l_elm_Glgrp := Elements(Glgrp); l_elm_MxChgrp := Elements(MxChgrp);
#####################################################
#Display("#Coset Decomposition Global/MaxChiral"); # #for debug
#####################################################
cd_Gl_MxC := CosetDecomposition(Glgrp, MxChgrp);
#Display(IsList(cd_Gl_MxC)); Display(cd_Gl_MxC); #for debug
#############################################
#Display("#Coset Decomposition Global/Local");# #for debug
#############################################
calcdegree := Size(Glgrp)/Size(Locgrp);
#Display(calcdegree); Display(calcdegree = DegCGr); #for debug
if calcdegree = DegCGr then
if isstabilizer = 1 then #harmonization
l_rep := []; cd_Gl_Loc := [];
for j in [1..DegCGr] do
#Print("####### j = ", j, "#########\n"); #for debug
l_rep[j] := RepresentativeAction(Glgrp, fixedpoint, j);
cd_Gl_Loc[j] := Elements(RightCoset(Locgrp, l_rep[j]));
od;
#Display(IsList(l_rep)); Display(l_rep); #for debug
#Display(IsList(cd_Gl_Loc)); Display(cd_Gl_Loc); #for debug
else #no harmonization
cd_Gl_Loc := CosetDecomposition(Glgrp, Locgrp);
#Display(IsList(cd_Gl_Loc)); Display(cd_Gl_Loc); #for debug
fi;
else #no harmonization
cd_Gl_Loc := CosetDecomposition(Glgrp, Locgrp);
#Display(IsList(cd_Gl_Loc)); Display(cd_Gl_Loc); #for debug
fi;
#################################################
#Display("#Coset Representation Global(/Local)");# #for debug
#################################################
s_perm_cd := [1..DegGr]; cosetrep := [];
for k in [1..Size(l_elm_Glgrp)] do
#Print("#### k:=", k, "#### \n"); #for debug
l_perm := cd_Gl_Loc*l_elm_Glgrp[k];
ll_perm := cd_Gl_MxC*l_elm_Glgrp[k];
#Display(l_elm_Glgrp[k]); #for debug
#Display(l_perm); #for debug
#Display(ll_perm); #for debug
perm_cd := [];
```

```
for j in [1..Size(cd_Gl_Loc)] do
for i in [1..Size(cd_Gl_Loc)] do
if IsEqualSet(cd_Gl_Loc[i],l_perm[j]) then
perm_cd[j] := i; break; fi;
od; od;
if DegCGr <> DegGr then
for j in [1..Size(cd_Gl_MxC)] do
for i in [1..Size(cd_Gl_MxC)] do
if IsEqualSet(cd_Gl_MxC[i],ll_perm[j]) then
perm_cd[DegCGr+j] := DegCGr+i; break; fi;
od; od;
fi;
#Display(perm_cd); #for debug
cosetrep[k] := PermListList(s_perm_cd, perm_cd);
#Display(cosetrep[k]); #for debug
od;
return cosetrep;
end; #end of CosetRepCF
##################################################
##################################################
## Function for detecting a row corresponding to #
## the local subgroup of globalgr(/localgrp) #
##################################################
detectTomSubgroup := function(globalgrp,localgrp,maxchgrp,degree,degreefull)
#detectTomSubgroup := function(Glgrp,Locgrp,MxChgrp,DegCGr,DegGr)
local i, j, k,
#Glgrp, Locgrp, MxChgrp, DegCGr, DegGr,
#l_elm_Glgrp, l_elm_MxChgrp,
tom_Glgrp,
l_conj_Locgrp, size_tom,
l_subsub, l_subsubX, l_marks;
Glgrp := globalgrp; Locgrp := localgrp; MxChgrp := maxchgrp;
DegCGr := degree; DegGr := degreefull;
#l_elm_Glgrp := Elements(Glgrp); l_elm_MxChgrp := Elements(MxChgrp);
l_conj_Locgrp := ConjugateSubgroups(Glgrp,Locgrp);
tom_Glgrp := TableOfMarks(Glgrp);
size_tom := Size(ConjugacyClassesSubgroups(Glgrp));
l_subsub := []; l_subsubX := []; l_marks := [];
for j in [1..size_tom] do
#Print("####### j = ", j, "#########\n"); #for debug
for i in [1..Size(l_conj_Locgrp)] do
if IsEqualSet(Elements(RepresentativeTom(tom_Glgrp,j)), Elements(l_conj_Locgrp[i]))
then
l_subsub := [j,i];
l_subsubX := [RepresentativeTom(tom_Glgrp,j),Locgrp];
l_marks := MatTom(tom_Glgrp)[j];
break;
fi;
od;
if l_subsub <> [] then break; fi;
od;
return [l_subsub,l_subsubX,l_marks];
end; #end of detectTomSubgroup
############################################################
############################################################
## function constructUSCICF #
## for construction of a USCI-CF #
## #
##To use this function solely, the following data #
##should be beforehand loaded: #
## (sample data of an octahedron of the point group Oh) #
## DegCGr := 6; #degree #
## DegGr := 8; #full degree #
## Glgrp := Oh_octa; # Global group #
## (Locgrp := C4v_octa; # Local group) not required #
## MxChgrp := O_octa; # Maximal chiral subgroup #
## tom_Glgrp := tom_Oh_octa; # tom of Glgrp #
## row_list := (vector of fixed points for Glgrp(/Locgrp) #
##If not, it will ended with the return value 'false'. #
############################################################
row_list :=[]; #to be replaced by a suitable list
```

```
constructUSCICF := function(subgroup_no)
local i, j, k, ii, kk,
#Glgrp, Locgrp, MxChgrp, # to be given globally
#DegCGr, DegGr, row_list, # to be given globally
#tom_Glgrp, # to be given globally
size_tom, alignsub_group, alignsubX_group,
calclist_subGr, l_AchOrCh,
InnSubGr, InnSubGr_list, sizeInnSubGr_list,
submark, submarkno, sub_group, sub_groupA, sub_groupTemp,
tom_subgroup, mattom_subgroup,invmattom_subgroup, sorted,
InnSubSubGr,InnSubSubGr_list,
subsub_group, subsub_groupX,
subsublist, sublist, memb_subsublist, permX,
submarkX, row_subduction,
list_subduction, permutedSubTom, size_CR, transpinvmat,
tempSI, tempSIX, tempSIY, USCI_CF;
################################
# Check for an independent usage #
################################
#Print("#####",row_list, "####\n"); #for debug
#Print("######## Glgrp =", Glgrp, "########\n");
if Glgrp = Group([()]) then return false; fi;
if row_list = [] then return false; fi;
##############################################
# list for checking achiral or chiral subgroups #
##############################################
size_tom := Size(ConjugacyClassesSubgroups(Glgrp));
calclist_subGr := []; l_AchOrCh := [];
for i in [1..size_tom] do
calclist_subGr[i] := RepresentativeTom(tom_Glgrp,i);
if IsSubgroup(MxChgrp,calclist_subGr[i]) then
l_AchOrCh[i] := 1; #chiral
else
l_AchOrCh[i] := 2; #achiral
fi;
od;
#Display(calclist_subGr); #for debug
#Display(l_AchOrCh); #for debug
#############################################################
# Non-redundunt set of inner subgroups (SSG) of each subgroup #
#############################################################
#Display("#Inner subgroups of each subgroup"); #for debug
InnSubGr := SubsTom(tom_Glgrp);
#Display(InnSubGr); #for debug
#Display(Length(InnSubGr)); #for debug
#Display(IsList(InnSubGr)); #for debug
InnSubGr_list := InnSubGr[subgroup_no];
#Display(InnSubGr_list); #for debug
#Display(IsList(InnSubGr_list)); #for debug
sizeInnSubGr_list := Size(InnSubGr_list);
#Display(sizeInnSubGr_list); #for debug
#########################################################
# Construct a "tentative" row vector of subduced marks #
# Note: #
# A set of conjugate subgroups in Glgrp may be #
# divided into one or more sets of conjugate subgroups #
# in a subgroup to be considered. #
# Compare InnSubGr with InnSubSubGr. #
#########################################################
#Display("#Construct a row vector of subduced marks"); #for debug
submark := [];
for i in [1..sizeInnSubGr_list] do
submarkno := InnSubGr_list[i];
Add(submark,row_list[submarkno]);
od;
#Display(submark); #for debug
########################################
# Table of marks (tom) of a subgroup_no #
########################################
#Display("#Subgroup of subgroup_no"); #for debug
sub_group := RepresentativeTom(tom_Glgrp,subgroup_no);
#Display(sub_group); #for debug
#Display("#Tom of subgroup of subgroup_no, AsSubgroup"); #for debug
```

```
#Print("###sub_group =", sub_group, "####\n");
sub_groupA := AsSubgroup(Glgrp,sub_group);
#Print("###sub_groupA =", sub_groupA, "####\n");
tom_subgroup := TableOfMarks(sub_groupA);
#Display(tom_subgroup); #for debug
################################
# Inner subsubgroups of subgroup #
# Compare: #
# InnSubGr vs. InnSubSubGr #
################################
#Display("#Inner subsubgroups of subgroup"); #for debug
InnSubSubGr := SubsTom(tom_subgroup);
#Display(InnSubSubGr); #for debug
#Display(Length(InnSubSubGr)); #for debug
#Display(IsList(InnSubSubGr)); #for debug
InnSubSubGr_list := InnSubSubGr[Length(InnSubSubGr)];
#Display(InnSubSubGr_list); #for debug
##############################################
# Permutation of inner subsubgroups of subgroup #
# InnSubSubGr in accord with InnSubGr #
##############################################
#Display("#Calculation of Correspondence of InnSubSubGr to InnSubGr"); #for debug
subsublist:= InnSubSubGr_list;
sublist:= InnSubGr_list;
#Display(Size(sublist)); #for debug
#Display(Size(subsublist)); #for debug
alignsub_group := [];
alignsubX_group := [];
for i in [1..Size(subsublist)] do
#Display("#Subgroup of Sub_Gno"); #for debug
subsub_group := RepresentativeTom(tom_subgroup,subsublist[i]);
#Display(subsub_group); #for debug
subsub_groupX := AsSubgroup(Glgrp,subsub_group);
#Display(subsub_groupX); #for debug
for j in [1..Size(subsublist)] do
sub_groupTemp:= RepresentativeTom(tom_Glgrp,sublist[j]);
#Display(sub_groupTemp); #for debug
if IsEqualSet(ConjugateSubgroups(Glgrp,subsub_groupX), ConjugateSubgroups(Glgrp,sub_groupTemp)) then
Add(alignsub_group,sublist[j]);
Add(alignsubX_group,j);
break;
fi;
od; od;
#Display("#Correspondence of InnSubSubGr to InnSubGr"); #for debug
#Display(alignsub_group); Display(alignsubX_group); #for debug
#Display("#Calculation of permutation to be applied to Tom"); #for debug
for i in [2..Size(subsublist)] do
memb_subsublist := alignsubX_group[i];
#Display(memb_subsublist); #for debug
k := 0;
for j in [i+1..Size(subsublist)] do
if memb_subsublist = alignsubX_group[j] then
#membX_subsublist := memb_subsublist;
k := k+1;
fi;
od;
#Display("#Kvalue"); #for debug
#Display(k); #for debug
if k > 0 then
kk := 0;
for ii in [2..Size(subsublist)] do
if alignsubX_group[ii] = memb_subsublist then
kk := kk + 1;
alignsubX_group[ii] := alignsubX_group[ii] + kk-1;
elif alignsubX_group[ii] > memb_subsublist then
alignsubX_group[ii] := alignsubX_group[ii] + k;
fi;
#Display(alignsubX_group); #for debug
od;
fi;
od;
#Display("#Permuted subgroups of InnSubSubGr in accord with InnSubGr"); #for debug
#Display(alignsubX_group); #for debug
```

```
#Display("#Permutation to be applied to Tom of InnSubSubGr in accord with InnSubGr"); #for debug
permX := PermListList(subsublist,alignsubX_group);
#Display(permX); #for debug
#Display("#SSG of the subgroup subgroup_no"); #for debug
permutedSubTom := Permuted(alignsub_group,permX);
#Display(permutedSubTom); #for debug
#Display("#validity of the SSG of the subgroup SubGno"); #for debug
#Display(InnSubGr_list); #for debug
#Display(InnSubGr_list = permutedSubTom); #for debug
#Display(IsEqualSet(InnSubGr_list, permutedSubTom)); #for debug
#####################################################################
# Sorted Tom of sub_group due to the allignment of the orignal group #
#####################################################################
#Display("#Sorted Tom of sub_group due to the allignment of the orignal group"); #for debug
sorted := SortedTom(tom_subgroup, permX);
#Display(sorted); #for debug
###########################################################################
# Matrix of sorted Tom of sub_group due to the allignment of the orignal group #
###########################################################################
#Display("#Matrix form of Tom of sub_group"); #for debug
mattom_subgroup := MatTom(sorted);
#Display(mattom_subgroup); #for debug
#################################################################################
# Inverse Matrix of sorted Tom of sub_group due to the allignment of the orignal group #
#################################################################################
#Display("#Inverse matrix form of Tom of sub_group"); #for debug
invmattom_subgroup := Inverse(mattom_subgroup);
#Display(invmattom_subgroup); #for debug
#Display("#Mark vector of the subgroup"); #for debug
#Display(row_list); #for debug
#Display("#Subgroups of the subgroup subgroup_no within Glgrp"); #for debug
#Display(InnSubGr_list); #for debug
#Display(submark); #for debug
#Display("#Modified allignment of subgroups of the subgroup subgroup_no within subgroup_no"); #for debug
#Display(permutedSubTom); #for debug
#Display("#Transposed matrix of Inverse matrix form of Tom of sub_group"); #for debug
transpinvmat := TransposedMat(invmattom_subgroup);
#Display(transpinvmat); #for debug
#################################################################################
# Calculation of Modified mark vector of the subgroup subgroup_no within subgroup_no #
#################################################################################
#Display("#Modified mark vector of the subgroup subgroup_no within subgroup_no"); #for debug
submarkX:=[];
for i in [1..Size(permutedSubTom)] do
submarkX[i] := row_list[permutedSubTom[i]];
od;
#Display(submarkX); #for debug
#########################################################
# Calculation of subduction vector of the subgroup SubGno #
#########################################################
#Display("#subduction vector of the subgroup subgroup_no"); #for debug
row_subduction := submarkX*invmattom_subgroup;
#Display(row_subduction); #for debug
#Display(DecomposedFixedPointVector(sorted,submarkX));#check #for debug
#Display("#subduction of global(/local) into subgroup_no"); #for debug
list_subduction:= [];
for i in [1..Size(row_subduction)] do
if row_subduction[i] > 0 then
Add(list_subduction, [row_subduction[i], subgroup_no, "/", permutedSubTom[i]]); #LLLLLL
fi;
od;
#Display(list_subduction); #for debug
#Display("#USCI-CF calculation"); #for debug
USCI_CF := 1;
for i in [1..Size(row_subduction)] do
tempSI := [];
if row_subduction[i] > 0 then
size_CR := Size(calclist_subGr[subgroup_no])/Size(calclist_subGr[permutedSubTom[i]]);
#Display(size_CR); #for debug
#Print("global =", subgroup_no, "; chiral or achiral= ", l_AchOrCh[subgroup_no], "\n"); #for debug
#Print("local =", permutedSubTom[i], "; chiral or achiral= ",
# l_AchOrCh[permutedSubTom[i]], "\n"); #for debug
if l_AchOrCh[subgroup_no] = 2 then
```

```
  if l_AchOrCh[permutedSubTom[i]] = 2 then
    tempSI := ["a_", size_CR]; #homospheric cycle
  elif l_AchOrCh[permutedSubTom[i]] = 1 then
     tempSI := ["c_", size_CR]; #enantiospheric cycle
  fi;
elif l_AchOrCh[subgroup_no] = 1 then
   tempSI := ["b_", size_CR]; #hemispheric
fi;
fi;
   tempSIX := JoinStringsWithSeparator(tempSI, "");
   tempSIY := Indeterminate(Rationals, tempSIX);
   USCI_CF := USCI_CF*tempSIY^row_subduction[i];
od;
return USCI_CF;
end; #end of constructUSCICF

##########################################
##########################################
## function: constructUSCICFlist #
## for construction of the list of USCIs #
## corresponding to Clgrp(/Locgrp): #
## e.g., #
## DegCGr := 6; #degree #
## DegGr := 8; #full degree #
## Glgrp := Oh_octa; # Global group #
## Locgrp := C4v_octa; # Local group #
##########################################
constructUSCICFlist := function(globalgrp,localgrp,maxchgrp,degree,degreefull)
#constructUSCICFlist := function(Glgrp,Locgrp,MxChgrp,DegCGr,DegGr)
local i,
#Glgrp, Locgrp, MxChgrp, DegCGr, DegGr, row_list,
l_USCICFs,
#tom_Glgrp,
size_tomX, subgrpTomGlLoc;
Glgrp := globalgrp; Locgrp := localgrp; MxChgrp := maxchgrp;
DegCGr := degree; DegGr := degreefull;
tom_Glgrp := TableOfMarks(Glgrp);
size_tomX := Size(ConjugacyClassesSubgroups(Glgrp));
#Display("#Calculation of a vector of fixed points"); #for debug
subgrpTomGlLoc := detectTomSubgroup(Glgrp, Locgrp, MxChgrp, DegCGr, DegGr);
#Display(subgrpTomGlLoc); #for debug
row_list := subgrpTomGlLoc[3]; #marks of the coset representation
#Display(row_list); #for debug
#Display("#Calculation of list of USCI-CFs"); #for debug
l_USCICFs := [];
for i in [1..size_tomX] do
#Print("######## i =", i, "########\n");
#Print("######## Glgrp =", Glgrp, "########\n");
l_USCICFs[i] := constructUSCICF(i);
od;
return l_USCICFs;
end; #end of the function constructUSCICFlist

##########################################
##########################################
## function: constructSCICFlist #
## for construction of the list of USCIs #
## corresponding to Clgrp(/Locgrp): #
## e.g., #
## DegCGr := 6; #degree #
## DegGr := 8; #full degree #
## Glgrp := Oh_octa; # Global group #
## row_list := [...]; #
##########################################
constructSCICFlist := function(globalgrp,maxchgrp,fixedpointvector,degree,degreefull)
#constructSCICFlist := function(Glgrp,MxChgrp,row_list,DegCGr,DegGr)
local i,
#Glgrp, Locgrp, MxChgrp, DegCGr, DegGr, row_list,
l_SCICFs,
#tom_Glgrp,
size_tom, subgrpTomGlLoc;
Glgrp := globalgrp;
```

```
MxChgrp := maxchgrp;
DegCGr := degree; DegGr := degreefull;
tom_Glgrp := TableOfMarks(Glgrp);
size_tom := Size(ConjugacyClassesSubgroups(Glgrp));
#Display("#Calculation of a vector of fixed points");
#subgrpTomGlLoc := detectTomSubgroup(Glgrp, Locgrp, MxChgrp, DegCGr, DegGr);
#Display(subgrpTomGlLoc);
##row_list := subgrpTomGlLoc[3]; #marks of the coset representation
row_list := fixedpointvector;
#Display(row_list); #for debug
#Display("#Calculation of list of SCI-CFs"); #for debug
l_SCICFs := [];
for i in [1..size_tom] do
l_SCICFs[i] := constructUSCICF(i);
od;
return l_SCICFs;
end; #end of the function constructSCICFlist

############################################################
# 2017/5/20 by Shinsaku Fujita #
############################################################
## function constructUSCITable #
## for construction of a USCI-CF table #
## #
## globalgrp (Glgrp) Global group #
## maxchgrp (MxChgrp) Maximum Chiral subgroup #
## num_gen Number of generators for subgrups #
## gen Generators for subgroups #
## degree Degree of a permutation group #
## degreefull Degree of a combined permutation group #
############################################################
constructUSCITable := function(globalgrp,maxchgrp,num_gen,gen,degree,degreefull)
local i, j, k,
l_subgroupGlgrp, l_cosetGlgrp,
templ_USCICFs,USCITable;
Glgrp := globalgrp;
MxChgrp := maxchgrp;
#
#List of Subgroups of Glgrp
l_subgroupGlgrp := []; #subgroup list
for i in [1..num_gen] do
l_subgroupGlgrp[i] := Subgroup(Glgrp,gen[i]);
od;
#Display(l_subgroupGlgrp);
#
#Use of the function 'detectTomSubgroup'
l_cosetGlgrp := [];
for i in [1..num_gen] do
l_cosetGlgrp[i] := detectTomSubgroup(Glgrp, l_subgroupGlgrp[i], MxChgrp, 6, 8);
od;
#Display(l_cosetGlgrp);
#####USCI-CFs#####
templ_USCICFs := [];
for i in [1..num_gen] do
templ_USCICFs[i] := constructUSCICFlist(Glgrp, l_subgroupGlgrp[i], MxChgrp, 6, 8);
od;
#####USCI Table####
USCITable := [];
for i in [1..num_gen] do
USCITable[i] := [];
for j in [1..num_gen] do
USCITable[i][j] := templ_USCICFs[i][l_cosetGlgrp[j][1][1]];
od;
od;
#Display(USCITable);
return USCITable;
end; #end of constructUSCITable
############################################################
# 2017/5/20 by Shinsaku Fujita #
############################################################
## function MarkTableforUSCI #
## corresponding to a USCI-CF table #
```

```
## due to function constructUSCITable #
## #
## globalgrp (Glgrp) Global group #
## maxchgrp (MxChgrp) Maximum Chiral subgroup #
## num_gen Number of generators for subgrups #
## gen Generators for subgroups #
## degree Degree of a permutation group #
## degreefull Degree of a combined permutation group #
############################################################
MarkTableforUSCI := function(globalgrp,maxchgrp,num_gen,gen,degree,degreefull)
local i, j, k,
l_subgroupGlgrp, l_cosetGlgrp,
templ_USCICFs,USCITable,
tempperm, permXtom, tom, sorted;
Glgrp := globalgrp;
MxChgrp := maxchgrp;
#
#List of Subgroups of Glgrp
l_subgroupGlgrp := []; #subgroup list
for i in [1..num_gen] do
l_subgroupGlgrp[i] := Subgroup(Glgrp,gen[i]);
od;
#Display(l_subgroupGlgrp);
#
#Use of the function 'detectTomSubgroup'
l_cosetGlgrp := [];
for i in [1..num_gen] do
l_cosetGlgrp[i] := detectTomSubgroup(Glgrp, l_subgroupGlgrp[i], MxChgrp, 6, 8);
od;
#Display(l_cosetGlgrp);
#
tempperm := [];
for i in [1..num_gen] do
tempperm[i] := l_cosetGlgrp[i][1][1];
od;
#Display(tempperm);
permXtom := PermListList(tempperm,[1..Size(tempperm)]);
#Display(permXtom);
#####Mark table#####
tom := TableOfMarks(Glgrp);
#Display(tom);
#####Sorted mark table####
sorted := SortedTom(tom, permXtom);
#Display(sorted);
return sorted;
end; #end of MarkTableforUSCI

############################################################
# 2017/6/5 by Shinsaku Fujita #
############################################################
## function constructSCICF #
## for construction of a list of SCI-CFs #
## #
## globalgrp (Glgrp) Global group #
## maxchgrp (MxChgrp) Maximum Chiral subgroup #
## USCITable USCI table corresponding to matrixtom #
## matrixtom, matrix form of a mark table #
## FPvector Fixed point vector #
## degree Degree of a permutation group #
## degreefull Degree of a combined permutation group #
############################################################
#constructSCICF := function(globalgrp,maxchgrp,matrixtom,USCITable,FPvector,degree,degreefull)
constructSCICF := function(globalgrp,maxchgrp,matrixtom,USCITable,FPvector)
local i, j, k, temp,
l_multi2, size_tom, USCI_Table, l_SCI_CF;
Glgrp := globalgrp;
MxChgrp := maxchgrp;
#list of multiplicities of coset representations
l_multi2 := FPvector*Inverse(matrixtom);
#Display(l_multi2); #for debug
#size of a mark table
size_tom := Size(ConjugacyClassesSubgroups(Glgrp));
```

```
#Display(size_tom); #for debug
#USCI table
USCI_Table := USCITable;
#calculation of SCI-CF
l_SCI_CF := [];
for i in [1..size_tom] do
   temp := 1;
 for j in [1..size_tom] do
  if l_multi2[j] = 0 then
  else
   temp := temp*USCI_Table[j][i]^l_multi2[j];
  fi;
 od;
 l_SCI_CF[i] := temp;
od;
#Display(l_SCI_CF); #for debug
return(l_SCI_CF);
end; #end of function constructSCICF

############################################################
# 2017/6/6 by Shinsaku Fujita #
############################################################
## function calculateFPvector #
## for a group derived by a given set of generators #
## #
## globalgrp (Glgrp) Global group #
## maxchgrp (MxChgrp) Maximum Chiral subgroup #
## num_gen Number of generators for subgrups #
## gen Generators for subgroups #
## degree Degree of a permutation group #
## degreefull Degree of a combined permutation group #
############################################################
calculateFPvector := function(globalgrp,maxchgrp,num_gen,gen,degree,degreefull)
local i, j, k,
size_tom, l_elements, l_fixedpoint, templist, size_group;
Glgrp := globalgrp;
MxChgrp := maxchgrp;
size_tom := Size(ConjugacyClassesSubgroups(Glgrp));
l_fixedpoint := [];
gen[1] := [()]; #debug 2017/5/7
for j in [1..size_tom] do
l_elements := Elements(Group(gen[j]));
size_group := Size(Group(gen[j]));
templist := [];
for i in [1..size_group] do
templist[i] := RestrictedPerm(l_elements[i], [1..degree]);
od;
#Display(templist); #for debug
#Display(NrMovedPoints(templist)); #for debug
l_fixedpoint[j] := degree - NrMovedPoints(templist);
od;
#Display(l_fixedpoint); #for debug
return(l_fixedpoint);
end; #end fo function calculateFPvector
```

Appendix B. `enum-tetra.gap` for Symmetry-Itemized Enumeration Based on a Tetrahedral Skeleton

The following souce code `enum-tetra.gap` aims at symmetry-itemized enumeration based on a tetrahedral skeleton, where the PCI method of Fujita's USCI approach is adopted. The file `enum-tetra.gap` is loaded by `Read` as shown in the first line (commented by #).

The PCI-CFs (`PCICF[1]`–`PCICF[11]`), which have been calculated in Source-Code 12, are copied and converted into the corresponding generating functions (`f_C1`–`f_Td`). The

function `calcCoeffGen` is used to evaluate the number of isomers with each composition (e.g., A^4), which appears as the coefficient of each monomial (e.g., A^4 corresponds to the partition [4,0,0,0,0,0,0,0,0,0,0,0]) in the generating functions derived from respective PCI-CFs (`f_C1`–`f_Td`). The function `calcCoeffGen` was defined in the file named `CICFgenCC.gapfunc` (Appendix A of Ref. [11]), which should be loaded at the first part of the source code. The results are collected in a tabular form (Table 1).

(`enum-tetra.gap`)

```
#Read("c:/fujita00/fujita2018/subductionTd/calcGAP3/enum-tetra.gap");
LogTo("c:/fujita00/fujita2018/subductionTd/calcGAP3/enum-tetralog.txt");

Read("c:/fujita00/fujita2018/subductionTd/calcGAP3/CICFgenCC.gapfunc"); #Loading of CICFgenCC.gapfunc

b_1 := Indeterminate(Rationals, "b_1"); b_2 := Indeterminate(Rationals, "b_2");
b_3 := Indeterminate(Rationals, "b_3"); b_4 := Indeterminate(Rationals, "b_4");
a_1 := Indeterminate(Rationals, "a_1"); a_2 := Indeterminate(Rationals, "a_2");
a_3 := Indeterminate(Rationals, "a_3"); a_4 := Indeterminate(Rationals, "a_4");
c_2 := Indeterminate(Rationals, "c_2"); c_4 := Indeterminate(Rationals, "c_4");

PCICF := [ ];
PCICF[1] := 1/24*b_1^4-1/4*a_1^2*c_2-1/6*b_1*b_3-1/8*b_2^2+1/2*a_1*a_3+1/4*a_2^2+1/4*b_4-1/2*a_4;
PCICF[2] := 1/4*b_2^2-1/4*a_2^2-1/4*b_4+1/2*a_4-1/4*c_4;
PCICF[3] := 1/2*a_1^2*c_2-a_1*a_3-1/2*a_2^2+a_4;
PCICF[4] := 1/2*b_1*b_3-1/2*a_1*a_3-1/2*b_4+1/2*a_4;
PCICF[5] := -1/2*a_4+1/2*c_4;
PCICF[6] := 0;
PCICF[7] := 1/2*a_2^2-1/2*a_4;
PCICF[8] := a_1*a_3-a_4;
PCICF[9] := 0;
PCICF[10] := 1/2*b_4-1/2*a_4;
PCICF[11] := a_4;

A := Indeterminate(Rationals, "A"); B := Indeterminate(Rationals, "B");
C := Indeterminate(Rationals, "C"); D := Indeterminate(Rationals, "D");
p := Indeterminate(Rationals, "p"); P := Indeterminate(Rationals, "P");
q := Indeterminate(Rationals, "q"); Q := Indeterminate(Rationals, "Q");
r := Indeterminate(Rationals, "r"); R := Indeterminate(Rationals, "R");
s := Indeterminate(Rationals, "s"); S := Indeterminate(Rationals, "S");

 aa_1 := A + B + C + D;
 aa_2 := A^2 + B^2 + C^2 + D^2;
 aa_3 := A^3 + B^3 + C^3 + D^3;
 aa_4 := A^4 + B^4 + C^4 + D^4;
 bb_1 := A + B + C + D + p + q + r + s + P + Q + R + S;
 bb_2 := A^2 + B^2 + C^2 + D^2 + p^2 + q^2 + r^2 + s^2 + P^2 + Q^2 + R^2 + S^2;
 bb_3 := A^3 + B^3 + C^3 + D^3 + p^3 + q^3 + r^3 + s^3 + P^3 + Q^3 + R^3 + S^3;
 bb_4 := A^4 + B^4 + C^4 + D^4 + p^4 + q^4 + r^4 + s^4 + P^4 + Q^4 + R^4 + S^4;
 cc_2 := A^2 + B^2 + C^2 + D^2 + 2*p*P + 2*q*Q + 2*r*R + 2*s*S;
 cc_4 := A^4 + B^4 + C^4 + D^4 + 2*p^2*P^2 + 2*q^2*Q^2 + 2*r^2*R^2 + 2*s^2*S^2;

f_C1 := Value(PCICF[1],
[a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4, c_2, c_4],
[aa_1, aa_2, aa_3, aa_4, bb_1, bb_2, bb_3, bb_4, cc_2, cc_4]);;

f_C2 := Value(PCICF[2],
[a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4, c_2, c_4],
[aa_1, aa_2, aa_3, aa_4, bb_1, bb_2, bb_3, bb_4, cc_2, cc_4]);;

f_Cs := Value(PCICF[3],
[a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4, c_2, c_4],
[aa_1, aa_2, aa_3, aa_4, bb_1, bb_2, bb_3, bb_4, cc_2, cc_4]);;

f_C3 := Value(PCICF[4],
[a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4, c_2, c_4],
```

```
[aa_1, aa_2, aa_3, aa_4, bb_1, bb_2, bb_3, bb_4, cc_2, cc_4]);;

f_S4 := Value(PCICF[5],
[a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4, c_2, c_4],
[aa_1, aa_2, aa_3, aa_4, bb_1, bb_2, bb_3, bb_4, cc_2, cc_4]);;

#f_D2 := Value(PCICF[6],
#[a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4, c_2, c_4],
#[aa_1, aa_2, aa_3, aa_4, bb_1, bb_2, bb_3, bb_4, cc_2, cc_4]);;

f_D2 := 0;

f_C2v := Value(PCICF[7],
[a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4, c_2, c_4],
[aa_1, aa_2, aa_3, aa_4, bb_1, bb_2, bb_3, bb_4, cc_2, cc_4]);;

f_C3v := Value(PCICF[8],
[a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4, c_2, c_4],
[aa_1, aa_2, aa_3, aa_4, bb_1, bb_2, bb_3, bb_4, cc_2, cc_4]);;

#f_D2d := Value(PCICF[9],
#[a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4, c_2, c_4],
#[aa_1, aa_2, aa_3, aa_4, bb_1, bb_2, bb_3, bb_4, cc_2, cc_4]);;

f_D2d := 0;

f_T := Value(PCICF[10],
[a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4, c_2, c_4],
[aa_1, aa_2, aa_3, aa_4, bb_1, bb_2, bb_3, bb_4, cc_2, cc_4]);;

f_Td := Value(PCICF[11],
[a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4, c_2, c_4],
[aa_1, aa_2, aa_3, aa_4, bb_1, bb_2, bb_3, bb_4, cc_2, cc_4]);;

list_partitions :=[];
calcCoeffGentetra := function(list_partitions)
local list_ligand_L, l_pp;
list_ligand_L := [A,B,C,D,p,P,q,Q,r,R,s,S];
l_pp := list_partitions;
Print("$", l_pp, "$ & ",
calcCoeffGen(f_C1, list_ligand_L, list_partitions), " & ",
calcCoeffGen(f_C2, list_ligand_L, list_partitions), " & ",
calcCoeffGen(f_Cs, list_ligand_L, list_partitions), " & ",
calcCoeffGen(f_C3, list_ligand_L, list_partitions), " & ",
calcCoeffGen(f_S4, list_ligand_L, list_partitions), " & ",
#calcCoeffGen(f_D2, list_ligand_L, list_partitions), " & ",
0, " & ",
calcCoeffGen(f_C2v, list_ligand_L, list_partitions), " & ",
calcCoeffGen(f_C3v, list_ligand_L, list_partitions), " & ",
#calcCoeffGen(f_D2d, list_ligand_L, list_partitions), " & ",
0, " & ",
calcCoeffGen(f_T, list_ligand_L, list_partitions), " & ",
calcCoeffGen(f_Td, list_ligand_L, list_partitions), " \\\\ \n");
end;

calcCoeffGentetra([4,0,0,0,0,0,0,0,0,0,0]);
calcCoeffGentetra([3,1,0,0,0,0,0,0,0,0,0]);
calcCoeffGentetra([3,0,0,0,1,0,0,0,0,0,0]);
calcCoeffGentetra([2,2,0,0,0,0,0,0,0,0,0]);
calcCoeffGentetra([2,0,0,0,2,0,0,0,0,0,0]);
calcCoeffGentetra([2,1,1,0,0,0,0,0,0,0,0]);
calcCoeffGentetra([2,1,0,0,1,0,0,0,0,0,0]);
calcCoeffGentetra([2,0,0,0,1,1,0,0,0,0,0]);
calcCoeffGentetra([2,0,0,0,1,0,1,0,0,0,0]);
calcCoeffGentetra([1,1,1,1,0,0,0,0,0,0,0]);
calcCoeffGentetra([1,1,1,0,1,0,0,0,0,0,0]);
calcCoeffGentetra([1,1,0,0,2,0,0,0,0,0,0]);
calcCoeffGentetra([1,1,0,0,1,1,0,0,0,0,0]);
calcCoeffGentetra([1,1,0,0,1,0,1,0,0,0,0]);
calcCoeffGentetra([1,0,0,0,3,0,0,0,0,0,0]);
calcCoeffGentetra([1,0,0,0,2,1,0,0,0,0,0]);
calcCoeffGentetra([1,0,0,0,2,0,1,0,0,0,0]);
```

```
calcCoeffGentetra([1,0,0,0,1,1,1,0,0,0,0,0]);
calcCoeffGentetra([1,0,0,0,1,0,1,0,1,0,0,0]);
calcCoeffGentetra([0,0,0,0,4,0,0,0,0,0,0,0]);
calcCoeffGentetra([0,0,0,0,3,1,0,0,0,0,0,0]);
calcCoeffGentetra([0,0,0,0,3,0,1,0,0,0,0,0]);
calcCoeffGentetra([0,0,0,0,2,2,0,0,0,0,0,0]);
calcCoeffGentetra([0,0,0,0,2,1,1,0,0,0,0,0]);
calcCoeffGentetra([0,0,0,0,2,0,2,0,0,0,0,0]);
calcCoeffGentetra([0,0,0,0,2,0,1,1,0,0,0,0]);
calcCoeffGentetra([0,0,0,0,2,0,1,0,1,0,0,0]);
calcCoeffGentetra([0,0,0,0,1,1,1,1,0,0,0,0]);
calcCoeffGentetra([0,0,0,0,1,1,1,0,1,0,0,0]);
calcCoeffGentetra([0,0,0,0,1,0,1,0,1,0,1,0]);

LogTo();
```

References

[1] G. Pólya, R. C. Read, *Combinatorial Enumeration of Groups, Graphs, and Chemical Compounds*, Springer, New York, 1987.

[2] G. Pólya, Kombinatorische Anzahlbestimmungen für Gruppen, Graphen und chemische Verbindungen, *Acta Math.* **68** (1937) 145–254.

[3] S. Fujita, Chirality fittingness of an orbit governed by a coset representation. Integration of point-group and permutation-group theories to treat local chirality and prochirality, *J. Am. Chem. Soc.* **112** (1990) 3390–3397.

[4] S. Fujita, Promolecules for characterizing stereochemical relationships in non-rigid molecules, *Tetrahedron* **47** (1991) 31–46.

[5] S. Fujita, *Combinatorial Enumeration of Graphs, Three-Dimensional Structures, and Chemical Compounds*, Univ. Kragujevac, Kragujevac, 2013.

[6] S. Fujita, *Symmetry and Combinatorial Enumeration in Chemistry*, Springer, Berlin, 1991.

[7] S. Fujita, *Diagrammatical Approach to Molecular Symmetry and Enumeration of Stereoisomers*, Univ. Kragujevac, Kragujevac, 2007.

[8] S. Fujita, *Mathematical Stereochemistry*, De Gruyter, Berlin, 2015.

[9] https://www.gap-system.org/.

[10] S. Fujita, Computer-oriented representations of point groups and cycle indices with chirality fittingness (CI-CFs) calculated by the GAP system. Enumeration of three-dimensional structures of ligancy 4 by Fujita's proligand method, *MATCH Commun. Math. Comput. Chem.* **76** (2016) 379–400.

[11] S. Fujita, Computer-oriented representations of $\boldsymbol{O}_h$-skeletons for supporting combinatorial enumeration by Fujita's proligand method. GAP calculation of cycle indices with chirality fittingness (CI-CFs), *MATCH Commun. Math. Comput. Chem.* **77** (2017) 409–442.

[12] S. Fujita, Computer-oriented representations of *RS*-stereoisomeric groups and cycle indices with chirality fittingness (CI-CF) calculated by the GAP system. Enumeration of *RS*-stereoisomers by Fujita's proligand method, *MATCH Commun. Math. Comput. Chem.* **77** (2017) 443–478.

[13] S. Fujita, Systematic enumeration of high symmetry molecules by means of unit subduced cycle indices with and without chirality fittingness, *Bull. Chem. Soc. Jpn.* **63** (1990) 203–215.

[14] S. Fujita, Unit subduced cycle indices with and without chirality fittingness for $\boldsymbol{I}_h$ group. An application to systematic enumeration of dodecahedrane derivatives, *Bull. Chem. Soc. Jpn.* **63** (1990) 2759–2769.

[15] S. Fujita, Systematic enumerations of highly symmetric cage-shaped molecules by unit subduced cycle indices, *Bull. Chem. Soc. Jpn.* **62** (1989) 3771–3778.

[16] S. Fujita, Systematic classification of molecular symmetry by subductions of coset representations, *Bull. Chem. Soc. Jpn.* **63** (1990) 315–327.

[17] S. Fujita, Subduction of coset representations. An application to enumeration of chemical structures, *Theor. Chim. Acta* **76** (1989) 247–268.

[18] S. Fujita, Subduction of coset representations. An application to enumeration of chemical structures with achiral and chiral ligands, *J. Math. Chem.* **5** (1990) 121–156.

[19] S. Fujita, Enumeration of digraphs with a given automorphism group, *J. Math. Chem.* **12** (1993) 173–195.

[20] S. Fujita, Generalization of partial cycle indices and modified bisected mark tables for combinatorial enumeration, *Bull. Chem. Soc. Jpn.* **73** (2000) 329–339.

[21] S. Fujita, The USCI approach and elementary superposition for combinatorial enumeration, *Theor. Chim. Acta* **82** (1992) 473–498.

[22] S. Fujita, Symmetry-itemized enumeration of cubane derivatives as three-dimensional entities by the fixed-point matrix method of the USCI approach, *Bull. Chem. Soc. Jpn.* **84** (2011) 1192–1207.

[23] S. Fujita, Symmetry-itemized enumeration of cubane derivatives as three-dimensional entities by the partial-cycle-index method of the USCI approach, *Bull. Chem. Soc. Jpn.* **85** (2012) 793–810.

[24] S. Fujita, Symmetry-itemized enumeration of cubane derivatives as three-dimensional entities by the elementary-superposition method of the USCI approach, *Bull. Chem. Soc. Jpn.* **85** (2012) 811–821.

[25] S. Fujita, Integrated discussion on stereogenicity and chirality for restructuring stereochemistry, *J. Math. Chem.* **35** (2004) 265–287.

[26] S. Fujita, Graphs to chemical structures 1. Sphericity indices of cycles for stereochemical extension of Pólya's theorem, *Theor. Chem. Acc.* **113** (2005) 73–79.