Basics

In abstract algebra, the commutator subgroup or derived subgroup of a group is the subgroup generated by all the commutators of the group.

For elements $g$ and $h$ of a group $G$, one of the two expressions of the commutator of $g$ and $h$ is $[g,h] = ghg'h'$. The commutator $[g,h]$ is equal to the identity element $e$ if and only if $gh = hg$, that is, if and only if $g$ and $h$ commute. In general, $gh = [g,h]hg$.

However, the product of two or more commutators need not generally be a commutator.

An alternating group is the group of even permutations of a finite set. The alternating group on the set $\{1,\ldots,n\}$ is called the alternating group of degree $n$, or the alternating group on $n$ letters and denoted by $A_n$.

A Property of Alternating Groups

It has been proved by G. A. Miller that the alternating group on $n$ letters, $n \geq 5$, consists entirely of commutators. This was rediscovered over half a century later by O. Ore.

They demonstrated that any permutation $\sigma$ of $A_n$ can be written as the product of $2$ $n$-cycles $p_1$, $p_2$:

$$\sigma = p_1 \cdot p_2$$

Because permutation $p_2$ is an $n$-cycle, as $p_1$ and $p_1'$ are, and because permutations which show the same cycle structure are conjugates, there exist a permutation $\tau$ such that permutation $p_2$ is the conjugate of $p_1'$ by $\tau$:

$$p_2 = \tau \cdot p_1' \cdot \tau'$$

So that permutation $\sigma$ can now be written as a commutator:

$$\sigma = p_1 \cdot \tau \cdot p_1' \cdot \tau' = [p_1, \tau]$$

E. Bertram showed later that permutations $p_1$, $p_2$ need not be $n$-cycles, but $l$-cycles, where the necessary and sufficient condition on $l$ is:

$$(3n/4) \leq l \leq n$$
Commutator Example

The 24 facelets of a given orbit of corner-centers of a 7x7x7 cube can be uniquely identified from a set of 24 letters A…X. We can then define alternating group $A_{24}$ as the group of all even permutations of these letters.

Applied to $n = 24$, the general condition on $l$ gives:

$$18 \leq l \leq 24$$

We choose $l = 23$, $p_1 = [NR NL, NU R NB ND']^*$ and $\tau = (R NF' L' NF R')^*$, so that both $p_1$ and $p_2$ are 23-cycles.

Notice that algorithm [NR NL, NU R NB ND'] is a 'pure' 23-cycle, ie. it will not move or rotate any facelet that doesn't belong to the selected orbit of corner-centers.

By using CubeTwister, it can be shown that the composition of $p_1$ and $p_2$ gives a 5-cycle, which can be written either as the product:

$$\sigma = p_1 \cdot p_2 = [NR NL, NU R NB ND'] (R NF' L' NF R') [NR NL, NU R NB ND'] (R NF' L' NF R') (34 \text{ moves})$$

or as the commutator:

$$\sigma = [p_1, \tau] = [[NR NL, NU R NB ND'], (R NF' L' NF R')] (34 \text{ moves})$$

By using CubeTwister, it can be shown that the composition of $p_1$ and $p_2$ gives a 5-cycle, which can be written either as the product:

$$\sigma = p_1 \cdot p_2 = [NR NL, NU R NB ND'] (R NF' L' NF R') [NR NL, NU R NB ND'] (R NF' L' NF R') (34 \text{ moves})$$

or as the commutator:

$$\sigma = [p_1, \tau] = [[NR NL, NU R NB ND'], (R NF' L' NF R')] (34 \text{ moves})$$

Factorizing permutations into 2 cycles of length $l$ is generally not very efficient in terms of moves, though. We can search for a shorter algorithm using Super Cube Solver and compare solutions:

$$\sigma = (CKEFM) = NF SR ND NF2 ND' NF2 R' NF2 ND NF2 ND' L NF' (13 \text{ moves})$$

*SSE Notation
\[ p_1 = [NR \ \text{NL}, \ NU \ R \ NB \ ND'] \]

\[ p_2 = \tau \cdot p_1 \cdot \tau' = \tau (R \ \text{NF'} \ L' \ \text{NF} \ R') [NR \ \text{NL}, \ NU \ R \ NB \ ND'] (R \ \text{NF'} \ L' \ \text{NF} \ R')' \]
\[ \sigma = p_1 \cdot p_2 = [NR \ NL, \ NU \ R \ NB \ ND'] \ (R \ NF' \ L' \ NF \ R') \] 
\[ \sigma = [p_1, \tau] = [[NR \ NL, \ NU \ R \ NB \ ND'], \ (R \ NF' \ L' \ NF \ R')] \]
Semi-Commutators

A commutator is generally defined as: \([A, B] = AB·B'A'\), where A and B are sequences of moves, representing permutations of pieces on a cube.

It has been proved that \(A_n\), the alternating group on \(n\) items, \(n \geq 5\), consists entirely of commutators, so that any permutation of \(A_n\) can be represented by a commutator or by a product of commutators, where the two representations are strictly equivalent, at least from a theoretical standpoint.

It may, however, be of interest to find products of commutators that give short sequences of moves, at least for products of a few commutators. For this to happen, there must be move cancellation between consecutive commutators. If a maximum number of moves could be cancelled out this way, a commutator-like expression may eventually be obtained, which may be called a semi-commutator – a better name is still to be found, btw.

The structure of such a semi-commutator depends on a number of variables and on the direction of enclosing brackets. Notice that this is more of a personal notation than something to be widely used…

Examples below are given for 3 and 4 variables and can be easily extended to a higher number of variables.

3 variables

Expressions of semi-commutators

\([X, YZ] = X·YZ·X'·Z'Y'\) (commutator)
\([X, YZ] = X·YZ·X'·Y'Z'\) (semi-commutator)
\([XY, Z] = XY·Z·X'Y'·Z'\) (semi-commutator)
\([XY, Z] = [X, YZ]\)

Semi-commutators as products of 2 commutators (move cancellations shown in red)

\([X, YZ] = [X, YZ][Y, Z] = X·YZ·X'·Z'Y·Y·Z·Y'·Z' = X·YZ·X'·Z'Y'\)
\([XY, Z] = [X, Y][XY, Z] = X·Y·X'·Y'·YX·Z·X'Y'·Z' = XY·Z·X'Y'·Z'\)

Inverses of semi-commutators

\([X, YZ]' = [YZ, X]\)
\([XY, Z]' = [Z, XY]\)

4 variables

Expressions of semi-commutators

\([XY, ZP] = XY·ZP·YX'·P'Z'\) (commutator)
\([XY, ZP] = XY·ZP·YY'·ZP'\) (semi-commutator)
\([XY, ZP] = XY·ZP·XY'·P'Z'\) (semi-commutator)
\([XY, ZP] = XY·ZP·XY'·ZP'\) (semi-commutator)

Semi-commutators as products of 2 or 3 commutators (move cancellations shown in red)

\([XY, ZP] = [XY, ZP][Z, P] = X·Y·P·Y'·X·Z·P·Z'·P' = XY·ZP·YX'·Z'P'\)
\([XY, ZP] = [X, Y][XY, ZP] = X·Y·X'·Y·YY·ZP·X·Y·P'Z' = XY·ZP·X'Y'·P'Z'\)
\([XY, ZP] = [X, Y][XY, ZP][Z, P] = X·Y·X'·Y·YY·ZP·X·Y'·P'Z·Z·P' = XY·ZP·X'Y'·Z'P'\)

Inverses of semi-commutators

\([XY, ZP]' = [PZ, XY]\)
\([XY, ZP]' = [ZP, YX]\)
\([XY, ZP]' = [PZ, YX]\)

Semi-commutators may be included in a brute-force search, when searching for algorithms by sweeping variables that take values in a set of basic moves, until swept permutation and goal permutation match, like in this example, where 8 variables are used:

\([XYZPQ, AEG] = XYZPQ·AEG·Q'P'Z'Y'X'·G'E'A'\) (commutator)
\([XYZPQ, AEG] = XYZPQ·AEG·Q'P'Z'Y'X'·A'E'G'\) (semi-commutator)
\([XYZPQ, AEG] = XYZPQ·AEG·XY'ZP'Q'·G'E'A'\) (semi-commutator)
\([XYZPQ, AEG] = XYZPQ·AEG·XY'ZP'Q'·A'E'G'\) (semi-commutator)
**Semi-Commutator Example**

A short 5-cycle of edge-centers has been obtained from the following semi-commutator:

\[ R \text{ NU} , \ N3B' R' \text{ NU} R \ N3B' [ = R \text{ NU} N3B R' \text{ NU} R \ N3B' R' \text{ NU} N3B' R \text{ NU} R' N3B \ (14 \text{ moves}) \]

We can search for a shorter algorithm using [Super Cube Solver](http://www.randelshofer.ch/rubik/virtualcubes/vcube7/7x_scripts/7x_super_cube_solver/index_enVE.html) and compare solutions:

\[ [N3B', R \text{ NU} TR' F \text{ NR}] = N3B' R \text{ NU} TR' F \text{ NR} N3B NR' F' TR NU' R' \ (12 \text{ moves}) \]

Notice that move TR is the combination of moves R and NR, that is: TR = R NR, thus giving a shorter solution.

**SuperCube Solver – Edge-Centers 5-Cycle**

http://www.randelshofer.ch/rubik/virtualcubes/vcube7/7x_scripts/7x_super_cube_solver/index_enVE.html

---

**Scramble Algorithm**

\[ R \text{ NU} , \ N3B' R' \text{ NU} R \ N3B' [ = R \text{ NU} N3B R' \text{ NU} R \ N3B' R' \text{ NU} N3B' R \text{ NU} R' N3B \]

**Solution Algorithm**

\[ [N3B', R \text{ NU} TR' F \text{ NR}] = N3B' R \text{ NU} TR' F \text{ NR} N3B NR' F' TR NU' R' \]
Symmetric Commutators

Commutators (or semi-commutators) show structures that are symmetric in nature. If we consider, for example, the following commutator \([A, B]\) of 5 variables \([X, Y, Z, P, Q]\), written as:

\[
[A, B] = [X Y, Z P Q] = X Y · Z P Q · Y' X' · P' Z' = (X Y) · (Z P Q) · (X Y)' · (Z P Q)' = A · B · A' · B'
\]

we can see that the second half of this expression is simply the composition of the inverses of \(A\) and \(B\).

Knowing that a cube has a set of 48 symmetries, we can further expand the concept of this 'plain' commutator to the 'symmetric' commutator, where the inverses of \(A\) and \(B\) are replaced with \(A_s\) and \(B_s\), being the inverses of their respective transformations by any of the 48 cube symmetries, that is:

\[
[A, B]_s = [X Y, Z P Q]_s = X Y · Z P Q · X_s' Y_s' · Z_s' P_s' Q_s' = (X Y) · (Z P Q) · (X_s Y_s)' · (Z_s P_s Q_s)' = A · B · A_s' · B_s'
\]

In this notation, subscript 's' indicates that symmetry has been applied to the second half of the expression.

A plain commutator is then just a particular case of a symmetric commutator, for which the applied symmetry is simply the 'Identity' symmetry:

\[
F → F
R → R
U → U
L → L
D → D
B → B
\]

The concept of symmetric commutators can even be further expanded to symmetric semi-commutators as follows:

\[
[X Y, Z P Q]_s = X Y · Z P Q · X_s' Y_s' · Z_s' P_s' Q_s'
\]

It is already known that plain commutators work well in cases where only a few cube pieces are permuted. They are generally of less practical use for solving cube positions with many permuted pieces, though. But, if a scrambled cube shows a symmetric pattern, chances are good that a symmetric commutator could be found that may eventually solve it.

### Symmetric Commutator Examples

Symmetric commutators may be used instead of plain commutators in difficult cases, or for finding alternate (symmetric) solution algorithms to already known ones.

As an example, we will search for an alternate algorithm to the hardest distance-20 position of a 3x3x3 cube, using symmetric commutators.

According to the 'God's Number is 20' paper, the following position was the hardest to their programs to solve:

\[
F U' F 2 D' B U R' F' L D' R' U' L U B' D 2 R' F U 2 D 2
\]

The algorithm itself doesn't show any obvious symmetry, so we first have to search for symmetric cube positions, if any.

The position shows a symmetry about the \(F – B\) axis, so that a half-turn cube rotation (by move \(CF2\)) gives another position which is equivalent to the initial one.

Using Cube Explorer 5.00s, all optimal solutions to this algorithm, plus its 19 shifted versions, have been found. From the list, a 18-move algorithm was extracted that presents a symmetric commutator structure:

\[
R' L · D 2 U' F' L D U 2 F' · R' L · F D 2 U' R' F D U 2
\]

This algorithm can be rewritten as:
[R’ L, D2 U’ F’ L D U2 F’]s

where symmetry CF2 has been applied to the second half, as follows:

\[
\begin{align*}
F &\rightarrow F \\
R &\rightarrow L \\
U &\rightarrow D \\
L &\rightarrow R \\
D &\rightarrow U \\
B &\rightarrow B
\end{align*}
\]

Using Algorithm Finder Lite, this 18-move algorithm has been shifted, conjugated and transformed by symmetry to give the following symmetric solutions to the initial (unsymmetric) algorithm, where SR = R L':

\[
\begin{align*}
B' & D' L2 R F D' L' R2 F SU F' L2 R U F' L' R2 U B' \\
F' & L' U2 D F L' U' D2 F SR F' U2 D R F' U' D2 R F' \\
F & D' R2 L B D' R' L2 B SU B' R2 L U B' R' L2 U F' \\
B' & R' U2 D B R' U' D2 B SR' B' U2 D L B' U' D2 L B' \\
F' & U R2 L' B' U R L2 B' SU B R2 L D' B R L2 D' F \\
B' & R D2 U' B' R D U2 B' SR B D2 U' L' B D U2 L' B \\
B' & U L2 R' F' U L R2 F' SU F L2 R' D' F L R2 D' B \\
F' & L D2 U' F' L D U2 F' SR' F D2 U' R' F D U2 R' F
\end{align*}
\]

All solutions shown are 20-move algorithms in HTM and 19-move algorithms in STM.
Symmetric commutators may also be used for permuting a few cube pieces, although plain commutators will generally provide shorter solutions, as shown below for the case of corner-center 5-cycle.

<table>
<thead>
<tr>
<th>Symmetric Commutators</th>
<th>Commutators</th>
</tr>
</thead>
<tbody>
<tr>
<td>[NR, NU’ NR NU R][s (10 moves)</td>
<td>NF’ D NF ND’ NF’ D’ NF’ ND NF2 (9 moves)</td>
</tr>
<tr>
<td>[NR, ND’ NR’ ND D2][s (10 moves)</td>
<td>U D NR’ ND’ NR U’ D’ NR ND ND NR’ (10 moves)</td>
</tr>
</tbody>
</table>

**Symmetric Commutator – Example 2**

**7x7x7 Cube – Permutation (BIOLV) – Corner-Center 5-Cycle**

<table>
<thead>
<tr>
<th>[NR, NU’ NR NU R][s (s = CR’)]</th>
<th>[NR, ND’ NR’ ND D2][s (s = CU’)]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Position</td>
<td>Equivalent Position</td>
</tr>
<tr>
<td>NR NU’ NR NU R NR’ NF ND’ NF’ R’</td>
<td>CR’ NR NU’ NR NU R NR’ NF ND’ NF’ R’</td>
</tr>
</tbody>
</table>

**7x7x7 Cube – Permutation (HRMKT) – Corner-Center 5-Cycle**

<table>
<thead>
<tr>
<th>[NR, NU’ NR NU R][s (s = CR’)]</th>
<th>[NR, ND’ NR’ ND D2][s (s = CU’)]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Position</td>
<td>Equivalent Position</td>
</tr>
<tr>
<td>NR NU’ NR NU R NR’ NF ND’ NF’ R’</td>
<td>CR’ NR NU’ NR NU R NR’ NF ND’ NF’ R’</td>
</tr>
</tbody>
</table>

Cube Theory
Generalized Commutators

Any permutation of $A_n$ can be represented by a commutator or by a product of commutators, where the two representations are strictly equivalent, at least from a theoretical standpoint.

It may, however, be of interest to find products of commutators that give short sequences of moves, at least for products of a few commutators. For this to happen, there must be move cancellation between some consecutive commutators. In the case where there is move cancellation between all consecutive commutators in the product, we get a special case which may be considered as a generalized commutator, written as:

$$[X, Y] = X Y X' Y'$$
$$[X, Y, Z] = [X, Y] [Y, Z] = X Y X' Y' Y Z Y' Z' = X Y X' Z Y' Z'$$
$$[X, Y, Z, P] = [X, Y] [Y, Z] [Z, P] = X Y X' Y' Y Z Y' Z' Z P Z' P' = X Y X' Z Y' P Z' P'$$
$$[X, Y, Z, P, Q] = [X, Y] [Y, Z] [Z, P] [P, Q] = X Y X' Y' Y Z Y' Z' Z P Z' P' P Q Q' = X Y X' Z Y' P Z' Q P Q'$$