Deriving Macros for the Solving of a Rubik's Cube

Matt Neary

August 25, 2013

Abstract

Solving a Rubik's cube is undoubtedly non-trivial; however, with the right motivations, one can find algorithms that will manipulate the cube in useful ways, allowing one to gradually approach a solved cube. Using basic group theory we will analyze the permutation of facelets resultant of a face turn, and then the products of various permutation compositions. Having recognized the rearrangement of facelets, we can identify the area of manipulation and work toward matching our desired region of change to that area.

1 Notation

In our discussion of the Rubik's cube we will need a means of referring both (1) to the *facelets* by which the cube is composed, and (2) to the manipulations of the cube which are possible.

1.1 The State of a Cube

A 2x2 Rubik's cube consists of 6 faces, each with two layers of 2 cubes each. To identify a given facelet, i.e., a colored square on one of the six faces, we provide the following notation.

$$\begin{aligned} < face > &::= F \mid U \mid D \mid L \mid R \mid B \\ < edge > &::= U \mid D \\ < corner > &::= L \mid R \end{aligned}$$

$$< facelet > ::= < face > < edge > < corner >$$

This notation is sufficient to fully identify the state of a cube.

1.2 Manipulations of a Cube

Beyond identifying the state of a cube, we will further want a means of communicating manipulations of the cube, i.e., face-turns. There are 6 face- turn operations, one for each face of the cube. We will articulate each as a permutation of facelets.

$U :=$ (LUL FUL RUL BUL)(LUR FUR RUR BUR)(UUL UUR UDR UDL)
$D :=$ (LDL FDL RDL BDL)(LDR FDR RDR BDR)(DUL DUR DDR DDL)
$L :=$ (UUL FUL DUL BDR)(UDL FDL DDL BUR)(LUL LUR LDR LDL)
$R :=$ (UUR FUR DUR BDL)(UDR FDR DDR BUL)(RUL RUR RDR RDL)
$F :=$ (UDL RDL DUL LUR)(UDR RUL DUR LDR)(FUL FUR FDR FDL)
$B :=$ (UUL RUR DDL LDL)(UUR RDR DDR LUR)(BUL BUR BDR BDL)

Face-turns are the atomic manipulations of a cube. Howevever, much more complex manipulations are possible. We will later provide an inductive definition of the set of all manipulations, or permutations.

2 Groups

2.1 Definition of a Group

A group $\mathcal{G}$ consists of an underlying set and a binary operator on members of that set meeting the following criteria.

1. The operation is closed on $\mathcal{G}$, so for $h, g \in \mathcal{G}$, we have $h * g \in \mathcal{G}$.
2. The operation is asociatve, so for $h, g, f \in \mathcal{G}$, we have $(f*g)*h = f*(g*h)$.
3. There exists an identity element $\mathcal{I} \in \mathcal{G}$ such that $(\forall g \in \mathcal{G})(\mathcal{I} * g = g * \mathcal{I} = g)$.
4. Every element $g \in \mathcal{G}$ has an inverse relative to , g^{-1} such that $g * g^{-1} = g^{-1} * g = \mathcal{I}$.

2.2 Theorems of Groups

1. The identity element $\mathcal{I}$ is unique.
2. If $A * B = \mathcal{I}$, $A = B^{-1}$.
3. If $A * X = B * X$, $A = B$.
4. The inverse of $A * B$ is $B^{-1} * A^{-1}$,
5. $(A^{-1})^{-1} = \mathcal{I}$,

3 The Group of Cube Permutations

Obviously face-turns of a Rubik's cube are not commutative; if they were, any cube could be solved with no more than 2 turns of each face! This is the hallmark of groups and thus we have reason to believe that Rubik's cube manipulation would be well modeled by a group. We will begin by defining all possible cube manipulations, i.e., sequences of face-turns.

$$< \text{faceturn} > ::= U \mid D \mid L \mid R \mid F \mid B$$
$$< \text{permutation} > ::= < \text{faceturn} > \mid < \text{permutation} >< \text{faceturn} >$$

The set of all of these *permutations* translates well into a group which we will call $\mathcal{R}$, so long as we consider (1) AB to be the composition of permutations A and B, and (2) equality to be *extensive*, i.e., two permutations that bear the same manipulation to be equal. Now obviously this group represents all possible states of a cube, as face-turns and their composition are the only manipulations possible to perform on a cube, hold removing the stickers.

3.1 The Binary Operation on Cube Permutations

In our group $\mathcal{R}$, the binary operation will be the composition of two permutations. The operation is obviously closed on $\mathcal{R}$ given our prior inductive definition of permutations. Furthermore, is associative as there is no concept of grouping of Rubik's cube face-turns. The identity, $\mathcal{I}$, is simply the identity permutation, e.g., $RRRR$; recall our *extensive* interpretation of equality, its inclusion ensures that the identity is unique.

3.2 Inverses

All permutations A in the group $\mathcal{R}$ have an inverse denoted as A^{-1}. Recall from the theorems of groups that the inverse of a product over a group $A * B$ is equal to $B^{-1} * A^{-1}$. If you merely imagine performing two consecutive moves on a Rubik's cube, the reversed order of their inverse should be intuitive.

The inverse of a face-turn is that face-turn applied three times, or equivalently, in the opposite direction. This is trivial to show by working out the permutation compositions or simply by referencing the nature of cyclical permutations. For example, $\mathcal{I} = (\text{A B C D})^4 = (\text{A B C D})^3 * (\text{A B C D})$, and thus by Theorem 2, $(\text{A B C D})^{-1} = (\text{A B C D})^3$.

4 Subgroups of Cube Permutations

In our definition of $\mathcal{R}$, we defined a permutation as a composition of face-turns. Were we to restrict the face-turns of which they were composed, we would have made a *subgroup* of $\mathcal{R}$. The following will serve as an example.

$$< \text{faceturn} > ::= F^2R^2$$
$$< \text{permutation} > ::= < \text{faceturn} > \mid < \text{permutation} >< \text{faceturn} >$$

In this case we would call the set F^2R^2 the *generator* of the subgroup. This subgroup happens to have 6 elements, as you will find by repeating the move F^2R^2 until returning to the identity. The size of the generated subgroup always coincides with the cycle of the generator, i.e., the *order* of the permutation.

5 Permutations

We have already discussed the permutation of facelets caused by each face- turn. Now, having inductively defined the set of all permutations, we will discuss them on a more complex level.

5.1 Parity

We wish to define *parity* of a permutation as the number of 2-cycle permutations of which the permutation is composed. This can in turn be interpreted as the number of swapped facelets. We define a function for this purpose, *par*, as follows.

$$par(\text{A B}) = 1$$
$$par(\text{A } \cdots \text{ B}) = (par\ (\text{A } \cdots)) + 1$$

Given this inductive definition of parity, we can show that if a permutation has n elements, it will have parity $n - 1$. If the parity is odd, a permutation is said to be odd, and if even, even.

The parity of each face-turn is the sum of the parity of the two constituent permutations, i.e., $(4-1)+(4-1)=6$. Thus all face-turns are even permutations. Now since a permutation on the cube is defined inductively on face-turns, all permutations are even. Thus we will say each state of the cube has even parity.

We have shown all permutations on the cube to be of even parity. This means that no two facelets can be swapped without side-effects which will be of guidance in our attempts to solve the cube. We thus should expect cycles of length three, or *3-cycles*, to be significant in our solving of the cube.

6 Algorithms

In order to solve a Rubik's cube, our goal will be to find move sequences that perform a simple manipulation of minimal side-effects which we will call *algorithms*. The *support* of an algorithm will be the set of facelets or cubies permuted by its moves.

6.1 Methodology

The first layer of a 2x2 Rubik's cube can be solved quite easily with intuition. However, having solved the first layer it becomes clear that maintaining your progress as you attempt to finish the cube will not be an easy task. Ignoring the details of implementation, it should be clear that we aim to fulfill the following tasks.

1. Solve the First Layer
2. Orient the Bottom Layer
3. Permute the Bottom Layer

6.2 Solving the First Layer

In solving the first layer, the goal is to find permutations that move a bottom layer cubie in any orientation up to the top layer correctly oriented. Given the permutations brought about by the face-turns, and the insignificance of side-effects, this should not be too difficult. The key to keep in mind is that the support of a side turn excludes cubies on the opposite side. That may seem obvious, but may be looked over as a means of *caching* a top-colored cubie as you bring down the top layer, then allowing you to place the cubie and reset the side face.

6.3 Permuting the Bottom Layer

After the First Layer has been solved, you are probably struck by the variety of colors on the bottom face. We, however, will first address the position of cubies, rather than whether they are pointing in the right direction.

Hence our goal is to permute the bottom cubies with a support independent of the first layer. The most intuitive route to swapping corners would be to bring a back corner toward the front of last layer, *cache* it by rotating the last layer, and to then return the rotated side and top faces. This would be achieved by the following algorithm.

$$LU^{-1}L^{-1}U$$

Analysis of this move sequence shows the permutation to be the following.

(LUL LUR BUL UDL UUL FUL) (DDR BUR LDL UUR BDL RUR)

If we reduce the above to refer to the permutation of *cubies* we have a simpler view of the action performed, as follows.

(UDL UUL) (UUR DDR)

The UDL and UUL cubies were swapped as intended; however, there was the unintended side-effect swapping UUR with DDR. This side-effect extended the support of our move sequence to the first layer. Referencing the permutation listed above, we see that the side-effect originates in the UUR cubie.

To counter-act this side-effect we will look at the moves directing the cubie to the first layer. The second half of the move sequence, $L^{-1}U$ performs the following permutation, in terms of cubies.

(UDR UUR UUL) (DDR UDL DUR)

Thus prior to our resetting of the cube we should prepare the support that the DUR cubie will eventually end up in the DUR slot. DUR should go, therefore, to the UDL slot. The move sequence for this manipulation is quite obvious, and as follows.

$$R^{-1}U$$

Let's look at the permutation of our new move sequence, including the preparatory steps in the middle.

$$LU^{-1}R^{-1}UL^{-1}U = \text{(UUL DUL DDL UDL) (UUR UDR)}$$

Notice that once again we have undesirable side-effects, this time caused by our intermediate step. Regarding cubies, the issue can be resolved by bringing UDL to the DDL slot and DUL up to the last layer. This means aligning the last layer left above the first layer left, rotating the left side so that the two up edge cubies are on the bottom, and resetting the last layer. Thus we have the following final sequence to our algorithm.

$$U^2RU^2$$

Hence we have the following final algorithm, after combining U and U^2 into U^{-1}.

$$LU^{-1}R^{-1}UL^{-1}U^{-1}RU^2$$

Our final permutation is the following; beyond swapping the right cubies, it merely re-orients the left cubies. Recall that a swap without side-effects would be impossible, as discussed in our digression into parity.

(UUL LUL BUL) (RUL RUR FUR UUR UDR BUR) (FUL LUR UDL)

6.4 Orienting the Bottom Layer

With corners in their correct positions, our goal is now to correctly orient them while maintaining position. If the goal is to change orientation of the bottom pieces, the natural approach would be to rotate a side toward the bottom, *cache* the now rotated cubie, and then move back the side and bottom faces. Note that we will write all algorithms in the context of bottom-face facing up, i.e., U refers to a turn of the last layer. Our initial idea of an orientation algorithm would then translate into the following move sequence.

$$RU^2R^{-1}U^2$$

Note the need for the last layer to be turned twice to perform our desired caching. The performed permutation can be found to be as follows.

(FUL UUL FDR UDR BUR) (DUL FUR UUR UDL BUL) (RDL RUL RUR LUR LUL)

This may not be easily parsed. A summary of this permutation in terms of *cubies* is as follows.

(UUR UDL UUL DUL UDR)

Thus we have a *support* of {UUR, UDL, UUL, DUL, UDR}. This translates into an unwanted side-effect. To combat this we will aim to prepare the cube in a way such that this side-effect undoes a prior manipulation, essentially preemptively inverting the side-effect.

The most obvious means to prepare the support is to place in the UUL slot the DUL cubie; to do this we should simply bring DUL up to the last layer, *cache* it to the side, move back the side we brought up, then shift it from UDL to UUL. This translates into the following move sequence.

$$RUR^{-1}U$$

Putting together our support preparation with our last layer orientation we have a useful algorithm.

$$RUR^{-1}URU^2R^{-1}U^2$$

For the sake of completeness, here is the product of our two permutations.

(BUL UUL LUL) (UUR BUR RUR) (FUR UDR RUL)

As you can see, our algorithm now serves to change the orientation of UUL, UUR, and UDR.

7 Conclusion

We have successfully used basic group theory to design move patterns which will aid in solving a Rubik's cube. Most often these sequences of moves would be memorized, but you have seen that they can be just as well derived with a little logic, strategy, and permutation analysis.