Department of Mathematics
Técnico, Lisboa

Professor Roger Picken

Mathematics and Scottish Country Dance

Final Degree Project

Author:
Maria Eusébio - 83638

July 18, 2018

Contents

Introduction

One of the greatest challenges for a mathematician is to be able to explain to anyone that mathematics really can be found everywhere in our lives and it is much more than just numbers. So the easiest way to explain it is to relate it to something we truly enjoy.

Therefore, the choice of the theme for this project was quite natural for me, because I have been dancing ever since I can remember and dance is actually a part of me. Although I am more familiar with classical and contemporary dances I am interested in all kinds of dance and, since my mentor is a specialist in Scottish Country Dance (SCD), I decided to approach the mathematics in SCD.

But how can we find mathematics in SCD? Well, at an elementary level we can find mathematics in shapes, like lines, squares and circles (figures that the dancers form) and there are also regular patterns to the music, for instance the main patterns AABB and ABAB for the four 8-bar phrases making up a 32-bar tune.

And at a higher level, we can use matrices to describe the possible arrangements of dancers in a group of four after various figures; the changes to these arrangements can be thought of as geometric transformations, like reflections, rotations, and translations; or we can also think of permutations between the dancers. Besides actual mathematical content, the orderliness and repeated patterns are also characteristics of SCD that are related with mathematics.

In this project we can find some history and an explanation of SCD, some possible models that use the concepts mentioned above including one related to the purest of mathematics and finally our proposal of a new model that allows us to evaluate and analyze SCD.

Thus, the main goal of this project is to use a mathematical analysis in order to better understand SCD and to assess it in general, or in the case of a new dance, to improve it (besides understanding the relation between mathematics and dance).

Scottish Country Dancing

Country dances were originally danced in rural England in the 16th century, and gradually became adopted in wider classes of society. Country dance spread elsewhere and in France was called *contredanse*, reflecting also the position of the dancers facing their partners.

Country dances were introduced in Scotland in the 18th century, and developed in an independent direction through the addition of some new figures and accompanied by typically Scottish music. A number of other forms of country dancing are still practised nowadays, and include English country dancing, and contra dance.

Scottish country dancing (SCD) was almost dying out around the beginning of the 20th century, but fortunately was rescued by some dedicated individuals who formed a society to preserve this tradition. Through the efforts of this society, the Royal Scottish Country Dance Society (RSCDS), over many years, SCD ultimately became a worldwide movement.

The steps in SCD are energetic and in balletic style. Dances are performed in groups of couples called sets. By far the most common set arrangement is the longwise set, with a line of typically four couples facing their partner. The dancers execute a rich variety of figures, normally involving 2 or 3 couples dancing together, as well as other dance moves which are not classified as standard figures. In each repetition of the dance sequence a so-called progression occurs, which reorders the couples, and gradually brings each couple from the position where they started to the top or the bottom of the set and back again.

The repertoire of dances is vast, based on historical dances going back to the 18th century, supplemented by many new dances that have been devised following the same broad conventions. Counting only the dances published by the international organization (Royal Scottish Country Dance Society - RSCDS) there are already around 1000 dances, and many more dances are popular despite the lack of official recognition. The manual and other materials produced by the RSCDS have standardized dance instructions, dancing technique and teaching methods, so that SCD has become a truly international dance form.

The steps can be done in a quick time (reel or jig) or in a slow rhythm as a strathspey step. So we have essentially these five basic steps:

1. skip change of step
2. slipstep
3. pas de basque
4. strathspey travelling step
5. strathspey setting step

With only these five steps we can create lots of figures and then dances from the simplest to the most complex.

Braids, Dance and Maths

Our first approach was trying to construct a mathematical model for dances based on the *Braid Group*. Unfortunately, we found some constraints on this possible model, so we proceeded to another idea.

However, we are going to explain the mathematical notions of this general idea in order to understand the model and its flaws.

Definition 1. *A **path** in X is a smooth function $f : [0, 1] \to X$.*

Definition 2. *An **isotopy** of X is a smooth family of diffeomorphisms ϕ_t of X for $t \in [0, 1]$.*

Definition 3. *In $\mathbb{R}^3$, let A be the set of points with coordinates $y = 0$, $z = 1$, and $x = 1,2,3,...,n$, and B the set of points with $y = 0$; $z = 0$, and $x = 1,2,3,...,n$. An **n-strand braid** is a set of n non-intersecting smooth paths connecting the n points in A to the n points in B and such that the z component of the tangent vector to each strand is always negative. We say that two braids are equivalent if there is an isotopy of $\mathbb{R}^3$ which preserves each plane of constant z and takes one braid to the other.*

We can think of a braid as a collection of strands in space with fixed endpoints that are braided around each other. This lets us define the braid group.

Definition 4. *(geometric) The **n-strand braid group** consists of equivalence classes of braids under isotopies that preserve each plane of constant z, with the operation that connects the bottom of the first braid to the top of the second, and rescales so that the new braid is still unit length. The trivial braid has n parallel strands, and the inverse of a braid is its mirror image reflecting in the plane $z=0$.*

Definition 5. *(algebraic) The **n-strand braid group** is the group given by the generators $\{\sigma_1, ..., \sigma_{n-1}\}$ with the relations:*

$$\sigma_i \sigma_j = \sigma_j \sigma_i, \text{ for } |i - j| \geq 2$$

and

$$\sigma_i \sigma_{i+1} \sigma_i = \sigma_{i+1} \sigma_i \sigma_{i+1}, \text{ for } 1 \leq i \leq n - 2$$

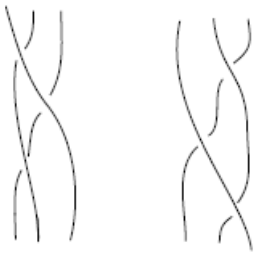

Figure 1: On the left we have $\sigma_i \sigma_{i+1} \sigma_i$ and on the right $\sigma_{i+1} \sigma_i \sigma_{i+1}$. There is clearly an isotopy between these two braids that only involves this segment of these three strands.

There is also another definition (topological), that is equivalent to the geometric and the algebraic definitions of the *Braid Group*, but we rely essentially on these two (because they were easier to understand and to relate with country dancing).

Thus, with these definitions, we can now relate the *Braid Group* with Country Dancing.

We considered that a $n-$strand braid represents a dance sequence, where the points are the n dancers, and the strands are the paths of each dancer. But the main problem of this model is that $\sigma\sigma^{-1} = e$ (where e is the trivial braid), which means that doing a sequence and then doing the inverse of that sequence is equivalent to doing nothing (which is absurd in terms of dancing). So we continued the search for a better model.

Representations of contra dance moves - Larry Copes

We will now see some of the possible notations for a mathematical representation of a contra dance, according to a study by the mathematician Larry Copes.

These three representations are suitable for only 2 couples (although we could add more couples, which would make the model more complicated) and show us only the changes of position that were made along the dance, so we can't evaluate it in the way that we wanted. However, they are useful to obtain a sequence where we interchange the two couples, starting with any position.

Permutation representation

Suppose that we have 4 members of a contra dance (2 couples) numbered like this:

3 4

2 1

For these positions we could indicate that the dancers in positions 1 and 3 switch places (somehow) by the notation (1 3). If we try to read that we would say that "the person in position 1 goes to position 3, and the person in position 3 goes to position 1".

Similarly, if everyone circled left 3/4 of the way around, we could write (1 4 3 2) and read "The dancer in position 1 goes to position 4, the dancer in position 4 goes to position 3, the dancer in position 3 goes to position 2 and the dancer in position 2 goes to position 1".

For a complete sequence that ends with everyone progressing, we need to determine a sequence that is equivalent to (1 2)(3 4), i.e., interchanging the dancers in positions 1 and 2, and also those in positions 3 and 4.

An advantage of this permutation notation is that we can see where a person moves over a sequence of figures.

Example 1. *Suppose we have the following sequence:*

(1 2)(3 4)

(1 3)(2 4)

(1 3)

no change

(1 4 3 2)

We can trace the path of the person who begins in location 1:

- *from position 1 to position 2*
- *from 2 to 4*
- *no change in the next two figures*
- *from 4 to 3*

Thus, we start describing the composite by writing: (1 3

Then we go on to consider what happens to the person who begins in position 3.

If we trace this all the way though, we see that this dancer ends up in position 1, so we close off our sequence: (1 3)

Next we do the same thing with the person in position 2 and in position 4 and we obtain the composite: (1 3)(2 4)

If we start to write a dance with these figures, we might ask what else is needed to finish it so that the final result is (1 2)(3 4) (to interchange the two couples).

Group element representation

We can also represent these numbers by letters that show how they are part of the group of symmetries of a square. One particular way of using letters helps us see relationships among the permutations:

- **I** (identity): permutation of figures, like lines forward and back, that don't change the locations of the dancers in the square
- **R**: permutation that rotates the square counterclockwise by 90 degrees ((1 4 3 2) in the previous notation)
- **R2**: rotation that rotates the square by 180 degrees (same as two **R** rotations; (1 3)(2 4) in the previous notation)
- **R3**: rotation through 270 degrees ((1 2 3 4) in the previous notation)
- **F** (flip): reflection of the square about a vertical line through the center of the horizontal sides ((1 2)(3 4) in the previous notation)

With these basic tools, we can write other possible permutations as a product of **F** with one of the **R**'s, for example:

FR is (2 4)

FR2 is (1 4)(2 3)

FR3 is (1 3)

In order to see what happens when one group element is followed by another, we write them next to each other and then simplify using the realization that:

- RF = FR3
- R2F = FR2
- FF = R4 = I

Then, for example, we can see that:

- FR2F = FFR = IR = R
- FRFR2 = FFR3R2 = FFR4R = IIR = R
- ($FRFR2 \neq$)FR2FR = FFR2R = IR3 = R3

So we have to worry about the order in which elements are combined and we make the following multiplication table:

	I	R	R2	R3	F	FR	FR2	FR3
I	I	R	R2	R3	F	FR	FR2	FR3
R	R	R2	R3	I	FR3	F	FR	FR2
R2	R2	R3	I	R	FR2	FR3	F	FR
R3	R3	I	R	R2	FR	FR2	FR3	F
F	F	FR	FR2	FR3	I	R	R2	R3
FR	FR	FR2	FR3	F	R3	I	R	R2
FR2	FR2	FR3	F	FR	R2	R3	I	R
FR3	FR3	F	FR	FR2	R	R2	R3	I

The element listed in the left column comes first, and is followed by the element in the top row.

The table, once worked out, allows us to continue the example we began with the permutation notation. There we had gone through a sequence of figures to get to (1 3)(2 4), and we wanted to know what else to do to get to (1 2)(3 4).

In the notation of group elements, we had gotten to R2 and we want to get to F. Looking at the table, we find R2 in the left column and look to see what element across the top will give us F, which turns out to be FR2. So to finish the dance we must choose a figure or sequence of figures that result in FR2.

The properties of this table show that, in the terminology of abstract algebra, these elements form a group. It's called the *dihedral group of order eight*, symbolized by $D4$. This group is defined as the group of all 8 symmetries of the square. It has a cyclic subgroup comprising rotations (which is the cyclic subgroup generated by **R**) and has four reflections (the ones with an **F**) each being an involution: reflections about lines joining midpoints of opposite sides, and reflections about diagonals.

We must add that some advanced contra dance figures result in rearrangements that are not among the eight symmetries of a square.

For example, a half-figure-8 by the number 1 couple dancers results in either (2 4) or (1 3), depending on their starting position. Since squares are no longer squares if two adjacent vertices are interchanged, these rearrangements are not among the symmetries in $D4$.

Matrix representation

As the structure of the representation of the 2 couples at the beginning looks like a matrix, it is almost immediate to think that the the transformations can be represented by matrices as well, and that the result might be found by multiplying matrices in the standard way.

Indeed, matrices do represent linear transformations of the plane, but as such they operate on the coordinate plane, in which points (or vectors) are described by a pair of coordinates.

To that end, we can rename our four positions using coordinates:

(-1,1) **(1,1)**

(-1,-1) **(1,-1)**

And then we represent the entire square as a 4 x 2 matrix:

$$\begin{pmatrix} -1 & 1 \\ 1 & 1 \\ 1 & -1 \\ -1 & -1 \end{pmatrix}$$

This 4x2 matrix can be multiplied by a 2x2 matrix to get another 4x2 matrix, representing the new arrangement of dancers.

There are eight elementary matrices that contain 0's on one of the diagonals and 1 or -1 on the other. These eight matrices represent the eight permutations or group elements under consideration.

In the following table we can find these eight matrices and their relation to the other representations.

Permutation	**Group Element**	**Transformation Matrix**	**Examples of Typical Figures**
(1)(2)(3)(4)	**I**	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	lines forward and back, allemand once around
(1 2 3 4)	**R3**	$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$	circle right 3/4 (or left 1/4)
(1 3)(2 4)	**R2**	$\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$	right and left through, circle 1/2
(1 4 3 2)	**R**	$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$	circle left 3/4 (or right 1/4)
(1 2)(3 4)	**F**	$\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$	swing on side (if lady is to the left of gentleman)
(1 3)	**FR3**	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	ladies' chain (if ladies are in positions 1 and 3)
(1 4)(2 3)	**FR2**	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	California twirl, box the gnat
(2 4)	**FR**	$\begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$	gents allemande 1-1/2 (if gents are in positions 2 and 4)

Example 2. *If the transformation is represented by the matrix:*

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

then the product is:

$$\begin{pmatrix} -1 & 1 \\ 1 & 1 \\ 1 & -1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \\ -1 & -1 \\ -1 & 1 \end{pmatrix}$$

This means that:

- *the dancer in position (-1 1) moves to position (1, 1)*
- *the dancer in position (1, 1) moves to position (1, -1)*
- *the dancer in position (1, -1) moves to position (-1, -1)*
- *the dancer in position (-1, -1) moves to position (-1, 1)*

In the permutation representation this is (1 2 3 4) and in the group element representation this is the transformation $\boldsymbol{R}$.

Our Model: Dance Evaluator

It is very important that the dancers are comfortable with the choreography they are dancing and also that it causes a good impact in the eyes of the spectators. So, with this thought, we came up with the idea of creating a model (computational program) that evaluates a dance in terms of several components in order to assess it in general, or in the case of a new dance, to improve it and make it better for the dancers and for the people who watch it in a presentation.

The choice of the parameters was a bit subjective, because we do not really know which ones are the most important and relevant for this evaluation. However, we used our experience in dance to define some aspects that most aroused our curiosity and interest, as well as those that seem to have greatest importance. Therefore, the components that the program (function `eval`) will measure, in order to evaluate the dance, are:

- Balance
    1. use of hands
    2. rotations performed
- Balance between genders
- Effort
- Complexity

Each component is measured with a scale from 0 to 1 (0 to 100%) and some have also another measure with a scale from -1 to 1. Next, there is an explanation of each one.

Balance

The function `balance` is applied to a dancer and measures the relation between the use of right and left hands as well as the rotations in either direction. It also returns which hand is most used and in which direction the dancer rotates most.

Thus, `balance` returns a list of 2 sublists. The first sublist has the value of the balance related with the use of the hands (`balh`) and with the rotations (`balr`) performed, and the second sublist has the value of how much one hand is used more than the other, and how much more the dancer has rotated in one of the directions. The second sublist is useful to discover and later correct possible flaws (imbalances) in the sequence.

1. Hands

A dancer can use right, left or both hands while executing movements. The first element of the first sublist is a value that corresponds to **1** if the dancer uses each hand equally (balanced), i.e. the same number of times during the sequence, and to **0** if the dancer always uses the same hand (unbalanced). The first element of the second sublist is a value that corresponds to **1** if the dancer uses only the left hand and **-1** if the dancer uses only the right hand (the value **0** means that there is an equal use of both hands).

We find this relevant because a dancer should use equally both sides of the body in order to avoid injuries and imbalances in the body.

2. Rotations

A dancer can rotate to the left and to the right sides. The second element of the first sublist is a value that corresponds to **1** if the dancer performs an equal number of rotations to each side and **0** if the dancer only rotates to one side. The second element of the second sublist is a value that corresponds to **1** if the dancer rotates always to the left (counterclockwise) and **-1** if the dancer rotates always to the right (clockwise). The value **0** means that the dancer rotates equally in each direction.

This is an important factor because if the dancer has a balance of 100% he/she can avoid dizziness and have a better performance.

Balance between genders

The function `balgend` is applied to a sequence (or dance), resorts to the function `effort` (explained later), and measures the relation between the effort of female and male dancers and also how much more effort one makes compared to the other.

So, this function returns a list with the two values mentioned above. The first value of the list is **1** if the dance is totally balanced with regard to the effort of male and female dancers and **0** if it is unbalanced. The second value of the list is **1** if men make all the effort and **-1** if women make all the effort (the value **0** means they make the same effort), so it is useful to discover and later correct possible gender imbalances in the sequence.

In the 21^{st} century there is more and more talk about gender equality in all aspects of our lives, so dancing should not be an exception. Therefore, this component is important for greater gender harmony and equality.

Effort

The function `effort` is applied to a dancer and measures the effort a dancer makes while dancing. This effort is based on 4 parameters, which are:

- The number of times a dancer travels a greater than normal distance (`dis`); it takes more effort to travel a greater than normal distance
- The number of times a dancer moves (`mov`); it takes more effort to move than to stand
- The rotations that a dancer performs (`rot`); it takes more effort to perform more rotations (because of dizziness, for example)
- The number of times a dancer uses his/her hands (`hand`); it takes more effort to use the hands (because of the force exerted by the arm)

The value returned by this function is **1** if the dancer makes a maximum effort and **0** if the dancer makes no effort. This value is extracted from the following equation:

$$0.2 \times \texttt{dis} + 0.6 \times \texttt{mov} + 0.1 \times \texttt{rot} + 0.1 \times \texttt{hand}$$

Of course this is a very subjective choice of parameter weights and probably not the best one, however, this component is one of the most important because it is directly related with the well-being of the dancer, and therefore with his/her performance in the dance.

It has to be noticed that the *intensity* (see below for an explanation) is related to this component, in fact, it is the same thing as the parameter `mov`. Thus, whenever a dance has an intensity higher than 0%, the dancers make an effort greater than 0%, which is also greater than or equal to the intensity (because `effort` has 3 other parameters). Typically, if the intensity is 0%, the effort is also 0%, except in rare cases where a dancer moves only his/her hands and not his/her feet (e.g., clapping).

Intensity is the only previously known measure for evaluating these dances, and was our starting point for this model. The idea is to represent during how many bars each couple dances in each sequence of 8 bars and then give the percentage of intensity in one turn and in the whole dance. It appears in many descriptions of the dances in the SCD data base (SCDDB).

An example could be in the dance *The Chequered Court* (10787 in SCDDB):

800/844/888/866 = 70% (1 turn), 53% (whole dance)

In the second sequence of 8 bars, the dancers are performing the figure *Corners Pass and Turn*, where the first couple dances the whole time and the other two couples dance only for 4 bars (which is in accordance with our description in the **Appendices**).

The value 70% is obtained by adding all the numbers (8+16+24+20 = 68 in the example) and dividing by 96 (all couples dancing all the time).

Complexity

The function `complexity` is applied to a sequence and measures precisely the complexity of that sequence. This complexity is based on 3 parameters:

- The number of times the dancers move (`movf`); making movements is more complex than doing nothing (because we have to remember what the next step is and how do we do it)
- The number of times the dancers change the person they are interacting with (`pairf`); it is more complex to change more times the person that we are interacting with (because we have to remember who is next)
- The lack of uniformity of the dance (`mat[stepm]`); when everyone is doing different steps, the dance becomes more complex than when everyone is doing the same thing

The third parameter is the most complicated to calculate. First we build a matrix, `stepm` with the steps that each dancer (rows) does in each sequence of 2 bars (columns). Then we apply the auxiliary function `mat` that counts the number of all the different steps in each sequence of 2 bars and that in the end returns the lack of uniformity (`unif`) of the dance; this value is **1** if everyone does different steps in the whole dance and **0** if everyone does the same steps.

The value returned by `complexity` is **1** if the dance is as complex as possible and **0** if the dance is not complex at all. This value is extracted from the following equation:

$$0.2 \times \texttt{movf} + 0.4 \times \texttt{pairf} + 0.4 \times \texttt{mat[stepm]}$$

Again, the choice of parameter weights is very subjective.

This component is the one that makes the dance more interesting and appealing both for those who see it and for those who execute it.

Defining dancers and sequences

In order to apply all the functions explained above, we had to find the best way to define a dancer and a sequence. So we decided that a sequence is a list of the dancers that perform that sequence and a dancer is a list of sublists, where the first sublist is the initial information (`id`: number of the couple, `gend`: gender, `pos`: initial position) and the second sublist is another set of sublists, each representing the modifications made during a sequence of 2 bars. These modifications can be one of the following 5:

- {`"id",x`}: `x` is the number (initial `id`) of the person the dancer is interacting with; every time a dancer changes the person he/she is interacting with, this shift has to appear
- {`"pos",k`}: `k` is the number of the position in which he/she is at the end of the bar; this shift has to appear every time the dancer changes his/her position; in addition, there could be a third element ({`"pos",k,1`}), that is the value 1, if the dancer traveled a greater than normal distance
- {`"hand",h`}: `h` indicates the hand that is being used ("l": left, "r": right, "b": both); this must appear every time the dancer uses one or both hands while performing a step
- {`"rot",d`}: `d` indicates in which direction (positive values: counterclockwise, negative values: clockwise) the dancer performed the rotation and how much he/she turned (1 turn, 3/4 of a turn,...); this must appear every time the dancer performs a rotation
- {`"mov",s`}: `s` is the step (one of the 5 mentioned in the **Scottish Country Dancing** chapter) that the dancer is performing; this must appear every time a dancer moves

If there are no modifications in a sequence of 2 bars, an empty list, {}, must appear, representing those 2 bars.

Another Function: Concatenation

Since writing sequences, the way we defined it, is a laborious and time consuming task, we find it relevant to have a function that could concatenate sequences in order to build a larger sequence or even a complete dance.

Thus, the function `conc` is applied to 2 sequences (S and T, for example), resorts to the auxiliary function `finpos` (that returns the position of a dancer at the end of the sequence) and concatenates them if they have the same number of dancers and if the the final configuration of the first one (S) matches the initial configuration of the second one (T).

In case we want to concatenate more than one sequence we just need to apply iteratively `conc`. For example, if we want to concatenate the sequences S, T and U, we do `conc`$[$`conc`$[S, T], U]$.

Some Examples

After building the program we need to test it and try to make some useful observations. For that, next we present some examples using each function (the code is in the **Appendices**).

1. *A Highland Welcome* (2931 in SCDDB, **H** in the code) is a simple dance, so we constructed it by applying iteratively the function `conc` to the 4 sequences in which we divided the dance. We also applied the function `eval` and the result was `eval[H]` $= 0.797331$.

2. *Dance to Corners and Set* (**CS** in the code) is an example of a complex figure, because dancers are always changing the person they are interacting with, they perform 2 different steps and because of its lack of uniformity. When we applied the function `complexity` the result was `complexity[CS]` $= 0.566667$.

   If we compare with the complexity of *A Highland Welcome* (it has to be noted that we are comparing a sequence/figure with a whole dance), that is 0.325, we conclude that *Dance to Corners and Set* is more complex, which makes sense according to our intuition (*A Highland Welcome* is a very simple dance).

3. *Corners Pass and Turn* (**CT** in the code) is an example of a figure where some dancers stay still during some bars of the sequence, so they should make an effort lower than the dancers in *Dance to Corners and Set*, for example. And by applying the function `effort` to each dancer in both sequences we see that is true:

   `Map[effort,CS]` $= \{0.625, 0.625, 0.629167, 0.629167, 0.629167, 0.629167\}$

   `Map[effort,CT]` $= \{0.696875, 0.696875, 0.3375, 0.3375, 0.3375, 0.3375\}$

   We can see that the dancers from the second and third couples in **CT** make a lower (almost half) effort than the dancers in **CS**, and of course, the dancers from the first couple.

4. *Ladies' Chain* (**U** in the code) is an example of a figure where there is clearly an imbalance between genders, because the female dancers make a greater effort than the male dancers. This is due to female dancers travelling a greater than normal distance during some bars. So the program returns:

   `balgend[U]` $= \{0.929638, -0.0703625\}$

   This means that women make about 7% more effort than men. And if we test it with the dancers from the first couple (female and male, **R1** and **R2** respectively), the program returns:

   `effort[R1]` $= 0.784375$ and `effort[R2]` $= 0.68125$

5. *Rights and Lefts* (**T** in the code) is an example of a balanced figure in terms of the use of the hands, and if we apply the function `balance` to each dancer, the program returns:

   `Map[balance,T]` $= \{\{\{1,0\},\{0,1\}\},\{\{1,1\},\{0,0\}\},\{\{1,1\},\{0,0\}\},\{\{1,0\},\{0,1\}\}\}$

   This means that all dancers use equally both hands, but only the man from couple 1 and the woman from couple 2 rotate equally in each direction.

6. If we try to concatenate *Set and Rotate* (**S** in the code) with *Rights and Lefts* (note that we have chosen a specific initial configuration, but this could change) we see that we can do it in the order (**T**,**S**) but we can not do it in the order (**S**,**T**) (the code is in the **Appendices**), because the initial configuration of **T** is different from the final configuration of **S**, but the initial configuration of **S** is the same as the final configuration of **T**.

7. The three figures from **4.**, **5.** and **6.** are all simple figures, so we thought that they would have similar complexity values, but it turns out to be false:

   `complexity[S]` $= 0.2$

   `complexity[T]` $= 0.5$

   `complexity[U]` $= 0.45$

   But after a more careful analysis, these results make sense, because:

   - **S** is the only sequence where the dancers never change the person they are interacting with
   - In **T** the dancers always change the person they are interacting with
   - In **U** the women change the person they are interacting with, twice, but the men only change once

   Also, as the dancers are always moving and doing the same steps at the same time in each one of the three dances, the only difference between them is precisely the number of times the dancers change the person they are interacting with.

Conclusion

It was a pleasure to work on such an interesting project. We achieved most of the objectives, however, because we did not have an infinite amount of time some of them remain to be done, for example, a function that measures the space occupation.

Another issue that may be restrictive is the division of the bars into groups of 2. If we want to assess, for example, the figure *Grand Chain* (which occurs for instance in the dance *Lady Lucy Ramsay* - 3616 in SCDDB), we would have to do a division into single bars, or into groups of 2 bars as well as single bars. Although we can do that, it is a really laborious task and if we wanted to concatenate with another sequence we would have to rewrite that sequence (dividing the bars into single bars), because the modifications must be distributed in groups with the same number of bars (for a correct evaluation of the sequence or dance).

With this project we demonstrate that there is actually a relation between dance (in particular, SCD) and mathematics and that, with mathematics, it is possible to assess SCD and improve new dances that can be created.

We also had some ideas for someone interested to try to implement in the future:

- Add initial and final orientations for each dancer (where are they facing), in order to avoid abrupt changes of direction when concatenating sequences (try to achieve a good flow in the dance)
- Build a database with the main SCD figures, including a variety of initial and final configurations (in order to not waste time writing a sequence again each time and to make it easier to use the program)
- Carry out a study in order to define the weights of the components (that we defined in a subjective way), by making a survey amongst dancers or something like that

Acknowledgments

I would like to thank my mentor Professor Roger Picken for all his guidance and support in this project, my colleague Rodrigo Girão Serrão for all his patience and help with my programming questions (no matter how dumb they were) and finally my ballet teacher Cristina Lima Matos for arousing the passion for dance in me.

References

[1] http://www.larrycopes.com/contra/representations.html

[2] Rebecca Hoberg, Knots and Braids, University of Chicago, 2011

[3] http://my.strathspey.org/dd/index/

Appendices

Program

```
(* Definition of the dancer (P) and the sequence(S)*)
P = {
   {id, gend, pos},
   List[
    {{"id", x}, { "pos", k}, {"hand", h}, {"rot", d}, {"mov", s}},
    {}, (*no changes*)
    ...(*sequences of 2 bars*)
   ]};(*dancer:set of initial information and set of modifications over time*)
(*pos may have 2 arguments: the position in which he/she is at the end of the bar and,
    in addition, the value 1 for travel a greater than normal distance*)
(*id: number of the respective couple
gend: gender of the dancer
  pos: initial position of the dancer
    x: number (initial id) of the person the dancer is interacting with
    k: number of the respective position
      h: "l"-left hand,"r"-right hand,"b"-both hands
  d: negative values-clockwise, positive values-anticlockwise
  s: 1- skip change of step
     2- slip step
     3- pas de basque
     4- strathspey travelling step
     5- strathspey setting step
*)

info[P_] := P[[1]]; (*initial information: pair, gender, position*)
bars[P_] := P[[2]]; (*the changes during each sequence of 2 bars*)

S = {P1, P2, P3, P4} (*Sequence: set of dancers*)
```

```
(*Balance: measures the relation between the use of right and left hands
  and the rotations for each direction;
returns also which hand is used most and in which direction the dancer rotates most*)
balance = Function[P, Module[{c, r, rabs, i, j, p, balh, balr},
    c = {Count[bars[P], {"hand", "l"}, 2],
      Count[bars[P], {"hand", "r"}, 2]};
    (*c[[1]]-number of times the left hand is used, c[[2]]-number of times the right hand is used*)
    r = 0; (*counts the real value of the rotations*)
    rabs = 0; (*counts the absolute value of the rotations*)
    i = 1;
    While[i ≤ Length[bars[P]],
     j = 1;
     p = bars[P][[i]];
     While[j ≤ Length[p],
      If[First[p[[j]]] == "rot", r = r + p[[j, 2]]; rabs = rabs + Abs[p[[j, 2]]]];
      j++
     ];
     i++
    ];
    balh = 1 - (Abs[c[[1]] - c[[2]]]) / (c[[1]] + c[[2]]);
    balr = 1 - (Abs[r] / rabs);
    {{balh, balr}, {(c[[1]] - c[[2]]) / (c[[1]] + c[[2]]), r / rabs}}
    (*{1-equal use of the 2 hands, 0-only 1 hand is used;
     1-equal number of rotations to each side, 0-only rotations to one side},
    {1-uses only left hand,-1-uses only rigth hand;
     1-rotates always to left, -1-rotates always to right}*)
   ]
  ];


(*Balance of genders: measures the relation between the effort of female and male dancers
   and how much more effort one gender does compared to the other *)
balgend = Function[S,
   Module[{f, m, i, P},
    f = 0;
    m = 0;
    i = 1;
    While[i ≤ Length[S],
     P = S[[i]];
     If[info[P][[2]] == "f", f = f + effort[P], m = m + effort[P]];
     i++
    ];
    {1 - Abs[f - m] / (f + m), (m - f) / (f + m)}
    (*{1-totally balanced,0-unbalanced;
     0-they have the same effort, 1-men make all the effort, -1-women make all the effort}*)
   ]
  ];
```

```
(*Effort: measures the effort a dancer makes while dancing*)
effort = Function[P, Module[{mov, dis, rot, hand, p, i, j},
    dis = 0; (*counts the number of times a dancer travels a greater than normal distance*)
    mov = 0; (*counts the number of times a dancer moves*)
    rot = 0; (*counts the rotations that a dancer performs*)
    hand = 0; (*counts the number of times a dancer uses his/her hands*)
    i = 1;
    While[i ≤ Length[bars[P]],
     p = bars[P][[i]];
     j = 1;
     While[j ≤ Length[p],
      If[First[p[[j]]] == "pos" && Length[p[[j]]] == 3, dis++];
      If[First[p[[j]]] == "mov", mov++];
      If[First[p[[j]]] == "rot", rot = rot + Abs[p[[j, 2]]]];
      If[First[p[[j]]] == "hand" && (p[[j, 2]] == "r" || p[[j, 2]] == "l"), hand++];
      If[First[p[[j]]] == "hand" && p[[j, 2]] == "b", hand = hand + 2];
      j++
     ];
     i++
    ];
    dis = dis / Length[bars[P]];
    mov = mov / Length[bars[P]];
    rot = rot / Length[bars[P]];
    hand = hand / (2 * Length[bars[P]]);
    (0.20 * dis) + (0.60 * mov) + (0.10 * rot) + (0.10 * hand)
   ]
  ];
```

```
(*Complexity: measures the complexity of the sequence*)
complexity = Function[S, Module[{movf, pairf, stepm, mov, step, pair, k, i, j, p, d},
    movf = 0;
    pairf = 0;
    k = 1;
    stepm = {};
    While[k ≤ Length[S],
     d = S[[k]];
     mov = 0; (*counts the number of times a dancer moves*)
     step = {}; (*list of movements that a dancer makes*)
     pair = 0; (*counts the number of times a dancer changes the person he/she is interacting with*)
     i = 1;
     While[i ≤ Length[bars[d]],
      p = bars[d][[i]];
      j = 1;
      While[j ≤ Length[p],
       If[First[p[[j]]] == "mov", mov++; step = Append[step, p[[j, 2]]]];
       If[First[p[[j]]] == "id", pair++];
       j++
      ];
      If[Length[step] < i, step = Append[step, 0]]; (*when there is no movement*)
      i++
     ];
     movf = movf + (mov / Length[bars[d]]) * (1 / Length[S]);
     pairf = pairf + (pair / Length[bars[d]]) * (1 / Length[S]);
     stepm = Append[stepm, step];
     k++
    ];
    (0.4 * mat[stepm]) + (0.2 * movf) + (0.4 * pairf)
   ]];

(*auxiliary function to assess uniformity*)
mat = Function[M, Module[{i, j, unif, sin},
    unif = 0;
    i = 1;
    While[i ≤ Length[M[[1]]], (*columns/bars*)
     sin = {}; (*list of all the different steps that are executed*)
     j = 1;
     While[j ≤ Length[M], (*rows/people*)
      If[! MemberQ[sin, M[[j, i]]], sin = Append[sin, M[[j, i]]]];
      j++
     ];
     If[Length[M] ≤ 4,
      unif = unif + ((Length[sin] - 1) / (Length[M] - 1)) * (1 / Length[M[[1]]]),
      unif = unif + ((Length[sin] - 1) / 4) * (1 / Length[M[[1]]])
     ];
     i++
    ];
    unif (*1-everyone does different things, 0-everyone does the same thing*)
   ]];
```

```
(*Concatenation: concatenates sequences, creating larger dances*)
conc = Function[{S, T}, Module[{SS = S, TT = T, i, j, b},
    If[Length[S] == Length[T], (*if the number of dancers is different, we don't concatenate*)
     i = 1;
     b = True;
     While[i ≤ Length[S] && b,
      If[finpos[S[[i]]] ≠ info[T[[i]]][[3]], b = False];
      (*if the end of S and the beginning of T don't match, we don't concatenate*)
      i++
     ];
     If[b,
      j = 1;
      While[j ≤ Length[S],
       SS[[j, 2]] = Join[S[[j, 2]], T[[j, 2]]];
       (*concatenation of the modifications of each 2 dancers from S and T*)
       j++
      ];
      SS,
      Print["Different end and beginning!"]],
     Print["Different number of dancers!"]
    ]
   ]];

(*auxiliary function that gives the position of a dancer at the end of the sequence*)
finpos = Function[P, Module[{i, pos, p, j},
    pos = info[P][[3]];
    i = 1;
    While[i ≤ Length[bars[P]],
     p = bars[P][[i]];
     j = 1;
     While[j ≤ Length[p],
      If[First[p[[j]]] == "pos", pos = p[[j, 2]]];
      j++
     ];
     i++
    ];
    pos
   ]];
```

```
(*Evaluation of the dance in terms of balance, balance between genders, effort and complexity*)
eval = Function[S, Module[{i, d, bal, eff},
    i = 1;
    bal = 0; (*balance of the whole dance*)
    eff = 0; (*effort of the whole dance*)
    While[i ≤ Length[S],
     d = S[[i]];
     bal = bal + (balance[d][[1, 1]] / Length[S]) + (balance[d][[1, 2]] / Length[S]);
     eff = effort[d] / Length[S];
     i++
    ];
    (bal + balgend[S][[1]] + eff + complexity[S]) / 4
   ]];
```

A Highland Welcome

```
(*Hands round and back*)
H1 = {
        {1, "m", 1},
        List[
              {{"id", 1, 2}, {"pos", 4, 1}, {"hand", "b"}, {"rot", 0.5}, {"mov", 2}},
              {{"pos", 1, 1}, {"hand", "b"}, {"rot", 0.5}, {"mov", 2}},
              {{"pos", 4, 1}, {"hand", "b"}, {"rot", -0.5}, {"mov", 2}},
              {{"pos", 1, 1}, {"hand", "b"}, {"rot", -0.5}, {"mov", 2}}
    ]};
(*{"id",1,2} means the dancer is interacting with his partner and a dancer from couple 2*)
H2 = {
        {1, "f", 2},
        List[
              {{"id", 1, 2}, {"pos", 3, 1}, {"hand", "b"}, {"rot", 0.5}, {"mov", 2}},
              {{"pos", 2, 1}, {"hand", "b"}, {"rot", 0.5}, {"mov", 2}},
              {{"pos", 3, 1}, {"hand", "b"}, {"rot", -0.5}, {"mov", 2}},
              {{"pos", 2, 1}, {"hand", "b"}, {"rot", -0.5}, {"mov", 2}}
    ]};

H3 = {
        {2, "f", 3},
        List[
              {{"id", 1, 2}, {"pos", 2, 1}, {"hand", "b"}, {"rot", 0.5}, {"mov", 2}},
              {{"pos", 3, 1}, {"hand", "b"}, {"rot", 0.5}, {"mov", 2}},
              {{"pos", 2, 1}, {"hand", "b"}, {"rot", -0.5}, {"mov", 2}},
              {{"pos", 3, 1}, {"hand", "b"}, {"rot", -0.5}, {"mov", 2}}
    ]};

H4 = {
        {2, "m", 4},
        List[
              {{"id", 1, 2}, {"pos", 1, 1}, {"hand", "b"}, {"rot", 0.5}, {"mov", 2}},
              {{"pos", 4, 1}, {"hand", "b"}, {"rot", 0.5}, {"mov", 2}},
              {{"pos", 1, 1}, {"hand", "b"}, {"rot", -0.5}, {"mov", 2}},
              {{"pos", 4, 1}, {"hand", "b"}, {"rot", -0.5}, {"mov", 2}}
    ]};
SQ1 = {H1, H2, H3, H4};
```

```
(*Hands across and back*)
H5 = {
        {1, "m", 1},
        List[
              {{"id", 1, 2, 2}, {"pos", 4, 1}, {"hand", "r"}, {"rot", -0.5625}, {"mov", 1}},
              {{"pos", 1, 1}, {"hand", "r"}, {"rot", -0.5625}, {"mov", 1}},
              {{"pos", 4, 1}, {"hand", "l"}, {"rot", 0.5625}, {"mov", 1}},
              {{"pos", 1, 1}, {"hand", "l"}, {"rot", 0.5625}, {"mov", 1}}
    ]};
(*{"id",1,2,2} means the dancer is interacting with everyone*)
H6 = {
        {1, "f", 2},
        List[
              {{"id", 1, 2, 2}, {"pos", 3, 1}, {"hand", "r"}, {"rot", -0.4375}, {"mov", 1}},
              {{"pos", 2, 1}, {"hand", "r"}, {"rot", -0.4375}, {"mov", 1}},
              {{"pos", 3, 1}, {"hand", "l"}, {"rot", 0.4375}, {"mov", 1}},
              {{"pos", 2, 1}, {"hand", "l"}, {"rot", 0.4375}, {"mov", 1}}
    ]};

H7 = {
        {2, "f", 3},
        List[
              {{"id", 1, 1, 2}, {"pos", 2, 1}, {"hand", "r"}, {"rot", -0.4375}, {"mov", 1}},
              {{"pos", 3, 1}, {"hand", "r"}, {"rot", -0.4375}, {"mov", 1}},
              {{"pos", 2, 1}, {"hand", "l"}, {"rot", 0.4375}, {"mov", 1}},
              {{"pos", 3, 1}, {"hand", "l"}, {"rot", 0.4375}, {"mov", 1}}
    ]};

H8 = {
        {2, "m", 4},
        List[
              {{"id", 1, 1, 2}, {"pos", 1, 1}, {"hand", "r"}, {"rot", -0.5625}, {"mov", 1}},
              {{"pos", 4, 1}, {"hand", "r"}, {"rot", -0.5625}, {"mov", 1}},
              {{"pos", 1, 1}, {"hand", "l"}, {"rot", 0.5625}, {"mov", 1}},
              {{"pos", 4, 1}, {"hand", "l"}, {"rot", 0.5625}, {"mov", 1}}
    ]};
SQ2 = {H5, H6, H7, H8};
```

```
(*Turn opposite partner + partner*)
H9 = {
        {1, "m", 1},
        List[
              {{"id", 2}, {"pos", 2}, {"hand", "l"}, {"rot", 0.375}, {"mov", 1}},
              {{"pos", 1}, {"hand", "l"}, {"rot", 0.375}, {"mov", 1}},
              {{"id", 1}, {"pos", 2}, {"hand", "r"}, {"rot", -0.375}, {"mov", 1}},
              {{"pos", 1}, {"hand", "r"}, {"rot", -0.375}, {"mov", 1}}
    ]};

H10 = {
        {1, "f", 2},
        List[
              {{"id", 2}, {"pos", 1}, {"hand", "l"}, {"rot", 0.625}, {"mov", 1}},
              {{"pos", 2}, {"hand", "l"}, {"rot", 0.625}, {"mov", 1}},
              {{"id", 1}, {"pos", 1}, {"hand", "r"}, {"rot", -0.625}, {"mov", 1}},
              {{"pos", 2}, {"hand", "r"}, {"rot", -0.625}, {"mov", 1}}
    ]};

H11 = {
        {2, "f", 3},
        List[
              {{"id", 1}, {"pos", 4}, {"hand", "l"}, {"rot", 0.625}, {"mov", 1}},
              {{"pos", 3}, {"hand", "l"}, {"rot", 0.625}, {"mov", 1}},
              {{"id", 2}, {"pos", 4}, {"hand", "r"}, {"rot", -0.625}, {"mov", 1}},
              {{"pos", 3}, {"hand", "r"}, {"rot", -0.625}, {"mov", 1}}
    ]};

H12 = {
        {2, "m", 4},
        List[
              {{"id", 1}, {"pos", 3}, {"hand", "l"}, {"rot", 0.375}, {"mov", 1}},
              {{"pos", 4}, {"hand", "l"}, {"rot", 0.375}, {"mov", 1}},
              {{"id", 2}, {"pos", 3}, {"hand", "r"}, {"rot", -0.375}, {"mov", 1}},
              {{"pos", 4}, {"hand", "r"}, {"rot", -0.375}, {"mov", 1}}
    ]};
SQ3 = {H9, H10, H11, H12};
```

```
(*Advance and Retire + Arches*)
H13 = {
        {1, "m", 1},
        List[
              {{"id", 1, 2}, {"pos", 13}, {"hand", "r"}, {"mov", 1}},
     (*this position is between positions 1 and 3, nearer position 1*)
              {{"pos", 1}, {"hand", "r"}, {"mov", 1}},
              {{"pos", 3}, {"hand", "r"}, {"mov", 1}},
              {{"pos", 1}, {"hand", "r"}, {"mov", 1}}
     (*this position 1 is the position in the new configuration*)
    ]};

H14 = {
        {1, "f", 2},
        List[
              {{"id", 1, 2}, {"pos", 24}, {"hand", "l"}, {"mov", 1}},
              {{"pos", 2}, {"hand", "l"}, {"mov", 1}},
              {{"pos", 4}, {"hand", "l"}, {"mov", 1}},
              {{"pos", 2}, {"hand", "l"}, {"mov", 1}}
    ]};

H15 = {
        {2, "f", 3},
        List[
              {{"id", 1, 2}, {"pos", 31}, {"hand", "l"}, {"mov", 1}},
              {{"pos", 3}, {"hand", "l"}, {"mov", 1}},
              {{"pos", 1}, {"hand", "l"}, {"mov", 1}},
              {{"pos", 3}, {"hand", "l"}, {"mov", 1}}
    ]};

H16 = {
        {2, "m", 4},
        List[
              {{"id", 1, 2}, {"pos", 42}, {"hand", "r"}, {"mov", 1}},
              {{"pos", 4}, {"hand", "r"}, {"mov", 1}},
              {{"pos", 2}, {"hand", "r"}, {"mov", 1}},
              {{"pos", 4}, {"hand", "r"}, {"mov", 1}}
    ]};
SQ4 = {H13, H14, H15, H16};


(*A Highland Welcome*)
H = conc[conc[conc[SQ1, SQ2], SQ3], SQ4];
```

In[137]:= **H**

Out[137]= {{{1, m, 1}, {{{id, 1, 2}, {pos, 4, 1}, {hand, b}, {rot, 0.5}, {mov, 2}}, {{pos, 1, 1}, {hand, b}, {rot, 0.5}, {mov, 2}}, {{pos, 4, 1}, {hand, b}, {rot, -0.5}, {mov, 2}},
{{pos, 1, 1}, {hand, b}, {rot, -0.5}, {mov, 2}}, {{id, 1, 2, 2}, {pos, 4, 1}, {hand, r}, {rot, -0.5625}, {mov, 1}}, {{pos, 1, 1}, {hand, r}, {rot, -0.5625}, {mov, 1}},
{{pos, 4, 1}, {hand, l}, {rot, 0.5625}, {mov, 1}}, {{pos, 1, 1}, {hand, l}, {rot, 0.5625}, {mov, 1}}, {{id, 2}, {pos, 2}, {hand, l}, {rot, 0.375}, {mov, 1}},
{{pos, 1}, {hand, l}, {rot, 0.375}, {mov, 1}}, {{id, 1}, {pos, 2}, {hand, r}, {rot, -0.375}, {mov, 1}}, {{pos, 1}, {hand, r}, {rot, -0.375}, {mov, 1}},
{{id, 1, 2}, {pos, 13}, {hand, r}, {mov, 1}}, {{pos, 1}, {hand, r}, {mov, 1}}, {{pos, 3}, {hand, r}, {mov, 1}}, {{pos, 1}, {hand, r}, {mov, 1}}}},
{{1, f, 2}, {{{id, 1, 2}, {pos, 3, 1}, {hand, b}, {rot, 0.5}, {mov, 2}}, {{pos, 2, 1}, {hand, b}, {rot, 0.5}, {mov, 2}}, {{pos, 3, 1}, {hand, b}, {rot, -0.5}, {mov, 2}},
{{pos, 2, 1}, {hand, b}, {rot, -0.5}, {mov, 2}}, {{id, 1, 2, 2}, {pos, 3, 1}, {hand, r}, {rot, -0.4375}, {mov, 1}}, {{pos, 2, 1}, {hand, r}, {rot, -0.4375}, {mov, 1}},
{{pos, 3, 1}, {hand, l}, {rot, 0.4375}, {mov, 1}}, {{pos, 2, 1}, {hand, l}, {rot, 0.4375}, {mov, 1}}, {{id, 2}, {pos, 1}, {hand, l}, {rot, 0.625}, {mov, 1}},
{{pos, 2}, {hand, l}, {rot, 0.625}, {mov, 1}}, {{id, 1}, {pos, 1}, {hand, r}, {rot, -0.625}, {mov, 1}}, {{pos, 2}, {hand, r}, {rot, -0.625}, {mov, 1}},
{{id, 1, 2}, {pos, 24}, {hand, l}, {mov, 1}}, {{pos, 2}, {hand, l}, {mov, 1}}, {{pos, 4}, {hand, l}, {mov, 1}}, {{pos, 2}, {hand, l}, {mov, 1}}}},
{{2, f, 3}, {{{id, 1, 2}, {pos, 2, 1}, {hand, b}, {rot, 0.5}, {mov, 2}}, {{pos, 3, 1}, {hand, b}, {rot, 0.5}, {mov, 2}}, {{pos, 2, 1}, {hand, b}, {rot, -0.5}, {mov, 2}},
{{pos, 3, 1}, {hand, b}, {rot, -0.5}, {mov, 2}}, {{id, 1, 1, 2}, {pos, 2, 1}, {hand, r}, {rot, -0.4375}, {mov, 1}}, {{pos, 3, 1}, {hand, r}, {rot, -0.4375}, {mov, 1}},
{{pos, 2, 1}, {hand, l}, {rot, 0.4375}, {mov, 1}}, {{pos, 3, 1}, {hand, l}, {rot, 0.4375}, {mov, 1}}, {{id, 1}, {pos, 4}, {hand, l}, {rot, 0.625}, {mov, 1}},
{{pos, 3}, {hand, l}, {rot, 0.625}, {mov, 1}}, {{id, 2}, {pos, 4}, {hand, r}, {rot, -0.625}, {mov, 1}}, {{pos, 3}, {hand, r}, {rot, -0.625}, {mov, 1}},
{{id, 1, 2}, {pos, 31}, {hand, l}, {mov, 1}}, {{pos, 3}, {hand, l}, {mov, 1}}, {{pos, 1}, {hand, l}, {mov, 1}}, {{pos, 3}, {hand, l}, {mov, 1}}}},
{{2, m, 4}, {{{id, 1, 2}, {pos, 1, 1}, {hand, b}, {rot, 0.5}, {mov, 2}}, {{pos, 4, 1}, {hand, b}, {rot, 0.5}, {mov, 2}}, {{pos, 1, 1}, {hand, b}, {rot, -0.5}, {mov, 2}},
{{pos, 4, 1}, {hand, b}, {rot, -0.5}, {mov, 2}}, {{id, 1, 1, 2}, {pos, 1, 1}, {hand, r}, {rot, -0.5625}, {mov, 1}}, {{pos, 4, 1}, {hand, r}, {rot, -0.5625}, {mov, 1}},
{{pos, 1, 1}, {hand, l}, {rot, 0.5625}, {mov, 1}}, {{pos, 4, 1}, {hand, l}, {rot, 0.5625}, {mov, 1}}, {{id, 1}, {pos, 3}, {hand, l}, {rot, 0.375}, {mov, 1}},
{{pos, 4}, {hand, l}, {rot, 0.375}, {mov, 1}}, {{id, 2}, {pos, 3}, {hand, r}, {rot, -0.375}, {mov, 1}}, {{pos, 4}, {hand, r}, {rot, -0.375}, {mov, 1}},
{{id, 1, 2}, {pos, 42}, {hand, r}, {mov, 1}}, {{pos, 4}, {hand, r}, {mov, 1}}, {{pos, 2}, {hand, r}, {mov, 1}}, {{pos, 4}, {hand, r}, {mov, 1}}}}}

Corners Pass and Turn

```
C1 = {
       {1, "f", 22}, (*this position is in the middle, in the diagonal of position 2*)
       List[
            {{"id", 2}, {"pos", 2}, {"rot", -0.5}, {"mov", 1}},
            {{"pos", 66}, {"rot", -0.25}, {"mov", 1}},
            {{"id", 3}, {"pos", 6}, {"rot", -0.5}, {"mov", 1}},
            {{"pos", 3, 1}, {"rot", -0.625}, {"mov", 1}}
   ]};

C2 = {
       {1, "m", 55},
       List[
            {{"id", 3}, {"pos", 5}, {"rot", -0.5}, {"mov", 1}},
            {{"pos", 11}, {"rot", -0.25}, {"mov", 1}},
            {{"id", 2}, {"pos", 1}, {"rot", -0.5}, {"mov", 1}},
            {{"pos", 4, 1}, {"rot", -0.625}, {"mov", 1}}
   ]};
C3 = {
       {2, "m", 2},
       List[
            {{"id", 1, 3}, {"pos", 66}, {"hand", "r"}, {"rot", -0.625}, {"mov", 1}},
            {{"pos", 2}, {"rot", -0.375}, {"mov", 1}},
            {},
            {}

   ]};
C4 = {
       {2, "f", 1},
       List[
            {},
            {},
            {{"id", 1, 3}, {"pos", 22}, {"hand", "r"}, {"rot", -0.375}, {"mov", 1}},
            {{"id", 1}, {"pos", 1}, {"rot", -0.625}, {"mov", 1}}
   ]};
C5 = {
       {3, "f", 5},
       List[
            {{"id", 1, 2}, {"pos", 11}, {"hand", "r"}, {"rot", -0.625}, {"mov", 1}},
            {{"pos", 5}, {"rot", -0.375}, {"mov", 1}},
            {},
            {}
   ]};
C6 = {
       {3, "m", 6},
       List[
            {},
            {},
            {{"id", 1, 2}, {"pos", 55}, {"hand", "r"}, {"rot", -0.375}, {"mov", 1}},
            {{"id", 1}, {"pos", 6}, {"rot", -0.625}, {"mov", 1}}
   ]};

CT = {C1, C2, C3, C4, C5, C6};
```

Dance to Corners and Set

```
S1 = {
        {1, "f", 22}, (*this position is in the middle, in the diagonal of position 2*)
        List[
              {{"id", 2}, {"pos", 2}, {"rot", -0.5}, {"mov", 1}},
              {{"mov", 3}},
              {{"id", 2, 1}, {"pos", 55}, {"rot", -0.25}, {"mov", 1}},
              {{"id", 2}, {"pos", 6}, {"rot", -0.5}, {"mov", 1}},
              {{"mov", 3}},
              {{"id", 2, 1}, {"pos", 11}, {"rot", -0.25}, {"mov", 1}}
    ]};

S2 = {
        {1, "m", 55},
        List[
              {{"id", 3}, {"pos", 3}, {"rot", -0.5}, {"mov", 1}},
              {{"mov", 3}},
              {{"id", 3, 1}, {"pos", 22}, {"rot", -0.25}, {"mov", 1}},
              {{"id", 3}, {"pos", 1}, {"rot", -0.5}, {"mov", 1}},
              {{"mov", 3}},
              {{"id", 3, 1}, {"pos", 66}, {"rot", -0.25}, {"mov", 1}}
    ]};

S3 = {
        {2, "m", 2},
        List[
              {{"id", 1, 3}, {"pos", 55}, {"rot", -0.375}, {"mov", 1}},
              {{"id", 3}, {"pos", 6}, {"rot", -0.5}, {"mov", 1}},
              {{"mov", 3}},
              {{"id", 1, 3}, {"pos", 11}, {"rot", -0.25}, {"mov", 1}},
              {{"id", 3}, {"pos", 5}, {"rot", -0.5}, {"mov", 1}},
              {{"rot", 0.125}, {"mov", 3}}
    ]};
```

```
S4 = {
        {2, "f", 1},
        List[
              {{"rot", 0.125}, {"mov", 3}},
              {{"id", 3, 3}, {"pos", 66}, {"rot", -0.25}, {"mov", 1}},
              {{"id", 1}, {"pos", 2}, {"rot", -0.5}, {"mov", 1}},
              {{"mov", 3}},
              {{"id", 3, 3}, {"pos", 55}, {"rot", -0.25}, {"mov", 1}},
              {{"id", 1}, {"pos", 6}, {"rot", -0.625}, {"mov", 1}}
    ]};
S5 = {
        {3, "f", 5},
        List[
              {{"id", 1, 2}, {"pos", 22}, {"rot", -0.375}, {"mov", 1}},
              {{"id", 2}, {"pos", 1}, {"rot", -0.5}, {"mov", 1}},
              {{"mov", 3}},
              {{"id", 1, 2}, {"pos", 66}, {"rot", -0.25}, {"mov", 1}},
              {{"id", 2}, {"pos", 2}, {"rot", -0.5}, {"mov", 1}},
              {{"rot", 0.125}, {"mov", 3}}
    ]};
S6 = {
        {3, "m", 6},
        List[
              {{"rot", 0.125}, {"mov", 3}},
              {{"id", 2, 2}, {"pos", 11}, {"rot", -0.25}, {"mov", 1}},
              {{"id", 1}, {"pos", 5}, {"rot", -0.5}, {"mov", 1}},
              {{"mov", 3}},
              {{"id", 2, 2}, {"pos", 22}, {"rot", -0.25}, {"mov", 1}},
              {{"id", 1}, {"pos", 1}, {"rot", -0.625}, {"mov", 1}}
    ]};

CS = {S1, S2, S3, S4, S5, S6};
```

Ladies' Chain

```
R1 = {
        {1, "f", 1},
        List[
             {{"id", 2}, {"pos", 2, 1}, {"hand", "r"}, {"rot", -0.125}, {"mov", 1}},
             {{"id", 1}, {"pos", 4}, {"hand", "l"}, {"rot", 0.625}, {"mov", 1}},
             {{"id", 2}, {"pos", 3, 1}, {"hand", "r"}, {"rot", -0.125}, {"mov", 1}},
             {{"id", 2}, {"pos", 1}, {"hand", "l"}, {"rot", 0.5}, {"mov", 1}}
   ]};

R2 = {
        {1, "m", 2},
        List[
             {{"pos", 4}, {"rot", 0.25}, {"mov", 1}},
             {{"pos", 2}, {"hand", "l"}, {"rot", 0.5}, {"mov", 1}},
             {{"pos", 4}, {"rot", 0.5}, {"mov", 1}},
             {{"id", 2}, {"pos", 2}, {"hand", "l"}, {"rot", 1}, {"mov", 1}}
   ]};

R3 = {
        {2, "m", 3},
        List[
             {{"pos", 1}, {"rot", 0.25}, {"mov", 1}},
             {{"pos", 3}, {"hand", "l"}, {"rot", 0.5}, {"mov", 1}},
             {{"pos", 1}, {"rot", 0.5}, {"mov", 1}},
             {{"id", 1}, {"pos", 3}, {"hand", "l"}, {"rot", 1}, {"mov", 1}}
   ]};

R4 = {
        {2, "f", 4},
        List[
             {{"id", 1}, {"pos", 3, 1}, {"hand", "r"}, {"rot", -0.125}, {"mov", 1}},
             {{"id", 2}, {"pos", 1}, {"hand", "l"}, {"rot", 0.625}, {"mov", 1}},
             {{"id", 1}, {"pos", 2, 1}, {"hand", "r"}, {"rot", -0.125}, {"mov", 1}},
             {{"id", 1}, {"pos", 4}, {"hand", "l"}, {"rot", 0.5}, {"mov", 1}}
   ]};

U = {R1, R2, R3, R4};
```

Rights and Lefts

```
Q1 = {
        {1, "f", 1},
        List[
              {{"pos", 2}, {"hand", "r"}, {"rot", 0.25}, {"mov", 1}},
              {{"id", 2}, {"pos", 4}, {"hand", "l"}, {"rot", 0.25}, {"mov", 1}},
              {{"id", 1}, {"pos", 3}, {"hand", "r"}, {"rot", 0.25}, {"mov", 1}},
              {{"id", 2}, {"pos", 1}, {"hand", "l"}, {"rot", 0.25}, {"mov", 1}}
    ]};

Q2 = {
        {1, "m", 2},
        List[
              {{"pos", 1}, {"hand", "r"}, {"rot", -0.25}, {"mov", 1}},
              {{"id", 2}, {"pos", 3}, {"hand", "l"}, {"rot", -0.25}, {"mov", 1}},
              {{"id", 1}, {"pos", 4}, {"hand", "r"}, {"rot", -0.25}, {"mov", 1}},
              {{"id", 2}, {"pos", 2}, {"hand", "l"}, {"rot", 0.75}, {"mov", 1}}
    ]};

Q3 = {
        {2, "f", 3},
        List[
              {{"pos", 4}, {"hand", "r"}, {"rot", -0.25}, {"mov", 1}},
              {{"id", 1}, {"pos", 2}, {"hand", "l"}, {"rot", -0.25}, {"mov", 1}},
              {{"id", 2}, {"pos", 1}, {"hand", "r"}, {"rot", -0.25}, {"mov", 1}},
              {{"id", 1}, {"pos", 3}, {"hand", "l"}, {"rot", 0.75}, {"mov", 1}}
    ]};

Q4 = {
        {2, "m", 4},
        List[
              {{"pos", 3}, {"hand", "r"}, {"rot", 0.25}, {"mov", 1}},
              {{"id", 1}, {"pos", 1}, {"hand", "l"}, {"rot", 0.25}, {"mov", 1}},
              {{"id", 2}, {"pos", 2}, {"hand", "r"}, {"rot", 0.25}, {"mov", 1}},
              {{"id", 1}, {"pos", 4}, {"hand", "l"}, {"rot", 0.25}, {"mov", 1}}
    ]};

T = {Q1, Q2, Q3, Q4};
```

Set and Rotate

```
P1 = {
        {1, "f", 1},
        List[
              {{"rot", -0.5}, {"mov", 3}},
              {{"pos", 3}, {"rot", -0.75}, {"mov", 1}},
              {{"pos", 1}, {"hand", "r"}, {"rot", -0.5}, {"mov", 1}},
              {{"pos", 3}, {"rot", -0.25}, {"mov", 1}}
    ]};

P2 = {
        {1, "m", 2},
        List[
              {{"rot", -0.75}, {"mov", 3}},
              {{"pos", 1}, {"rot", -0.5}, {"mov", 1}},
              {{"pos", 3}, {"hand", "r"}, {"rot", -0.75}, {"mov", 1}},
              {{"pos", 4}, {"rot", -0.5}, {"mov", 1}}
    ]};

P3 = {
        {2, "f", 3},
        List[
              {{"rot", -0.75}, {"mov", 3}},
              {{"pos", 4}, {"rot", -0.5}, {"mov", 1}},
              {{"pos", 2}, {"hand", "r"}, {"rot", -0.25}, {"mov", 1}},
              {{"pos", 1}, {"rot", -0.5}, {"mov", 1}}
    ]};

P4 = {
        {2, "m", 4},
        List[
              {{"rot", -0.5}, {"mov", 3}},
              {{"pos", 2}, {"rot", -0.75}, {"mov", 1}},
              {{"pos", 4}, {"hand", "r"}, {"rot", -0.5}, {"mov", 1}},
              {{"pos", 2}, {"rot", -0.25}, {"mov", 1}}
    ]};

S = {P1, P2, P3, P4};
```

Some Examples

```
In[138]:= eval[H]
Out[138]= 0.797331

In[139]:= complexity[CS]
Out[139]= 0.566667

In[140]:= complexity[H]
Out[140]= 0.325

In[154]:= Map[effort, CS]
Out[154]= {0.625, 0.625, 0.629167, 0.629167, 0.629167, 0.629167}

In[155]:= Map[effort, CT]
Out[155]= {0.696875, 0.696875, 0.3375, 0.3375, 0.3375, 0.3375}

In[165]:= balgend[U]
Out[165]= {0.929638, -0.0703625}

In[167]:= {effort[R1], effort[R2]}
Out[167]= {0.784375, 0.68125}

In[173]:= Map[balance, T]
Out[173]= {{{1, 0.}, {0, 1.}}, {{1, 1.}, {0, 0.}}, {{1, 1.}, {0, 0.}}, {{1, 0.}, {0, 1.}}}


In[168]:= conc[S, T]
          Different end and beginning!

In[169]:= conc[T, S]
Out[169]= {{{1, f, 1}, {{pos, 2}, {hand, r}, {rot, 0.25}, {mov, 1}}, {{id, 2}, {pos, 4}, {hand, l}, {rot, 0.25}, {mov, 1}},
            {{id, 1}, {pos, 3}, {hand, r}, {rot, 0.25}, {mov, 1}}, {{id, 2}, {pos, 1}, {hand, l}, {rot, 0.25}, {mov, 1}}, {{rot, -0.5}, {mov, 3}},
            {{pos, 3}, {rot, -0.75}, {mov, 1}}, {{pos, 1}, {hand, r}, {rot, -0.5}, {mov, 1}}, {{pos, 3}, {rot, -0.25}, {mov, 1}}}},
           {{1, m, 2}, {{{pos, 1}, {hand, r}, {rot, -0.25}, {mov, 1}}, {{id, 2}, {pos, 3}, {hand, l}, {rot, -0.25}, {mov, 1}},
            {{id, 1}, {pos, 4}, {hand, r}, {rot, -0.25}, {mov, 1}}, {{id, 2}, {pos, 2}, {hand, l}, {rot, 0.75}, {mov, 1}}, {{rot, -0.75}, {mov, 3}},
            {{pos, 1}, {rot, -0.5}, {mov, 1}}, {{pos, 3}, {hand, r}, {rot, -0.75}, {mov, 1}}, {{pos, 4}, {rot, -0.5}, {mov, 1}}}},
           {{2, f, 3}, {{{pos, 4}, {hand, r}, {rot, -0.25}, {mov, 1}}, {{id, 1}, {pos, 2}, {hand, l}, {rot, -0.25}, {mov, 1}},
            {{id, 2}, {pos, 1}, {hand, r}, {rot, -0.25}, {mov, 1}}, {{id, 1}, {pos, 3}, {hand, l}, {rot, 0.75}, {mov, 1}}, {{rot, -0.75}, {mov, 3}},
            {{pos, 4}, {rot, -0.5}, {mov, 1}}, {{pos, 2}, {hand, r}, {rot, -0.25}, {mov, 1}}, {{pos, 1}, {rot, -0.5}, {mov, 1}}}},
           {{2, m, 4}, {{{pos, 3}, {hand, r}, {rot, 0.25}, {mov, 1}}, {{id, 1}, {pos, 1}, {hand, l}, {rot, 0.25}, {mov, 1}},
            {{id, 2}, {pos, 2}, {hand, r}, {rot, 0.25}, {mov, 1}}, {{id, 1}, {pos, 4}, {hand, l}, {rot, 0.25}, {mov, 1}}, {{rot, -0.5}, {mov, 3}},
            {{pos, 2}, {rot, -0.75}, {mov, 1}}, {{pos, 4}, {hand, r}, {rot, -0.5}, {mov, 1}}, {{pos, 2}, {rot, -0.25}, {mov, 1}}}}}

In[170]:= complexity[S]
Out[170]= 0.2

In[171]:= complexity[T]
Out[171]= 0.5

In[172]:= complexity[U]
Out[172]= 0.45
```