MTH 496
Senior Thesis

Adam M. Terwilliger
Department of Mathematics
Grand Valley State University
Allendale, MI 49501
terwilla@mail.gvsu.edu

Brian Drake
Department of Mathematics
Grand Valley State University
Allendale, MI 49501
drakeb@mail.gvsu.edu

Abstract

We extend previous work which defined a relation on the set of Dyck paths by defining a relation on the set of Schröder paths. We create a set of rules inspired by tilings that define a relation represented as a directed graph. Through implementation of these rules using Python, we can explore properties of digraphs on Schröder paths.

1 Background

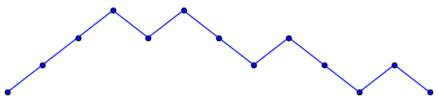
1.1 Related Work

Our work is inspired from work by Kenyon and Wilson (2011).

Kenyon and Wilson based their work upon Dyck paths motivated from Catalan numbers:

$$C_n = \frac{1}{n+1}\binom{2n}{n}$$

Figure 1: Example of Dyck Path



1.2 Definitions

Definition 1. A Dyck path is defined as a lattice path from $(0, 0)$ to $(2n, 0)$ consisting of n up steps of the form $(1, 1)$ and n down steps of the form $(-1, 1)$ which never goes below the x-axis $y = 0$.

Definition 2. A *Schröder path* is defined as a lattice path from $(0, 0)$ to $(2n, 0)$ consisting of some number of up steps of the form $(1, 1)$, down steps of the form $(-1, 1)$, and horizontal steps of the form $(2, 0)$, which never goes below the x-axis $y = 0$.

To every Schröder path, we assign a string of U, D and S. We use such strings to create the relation as follows.

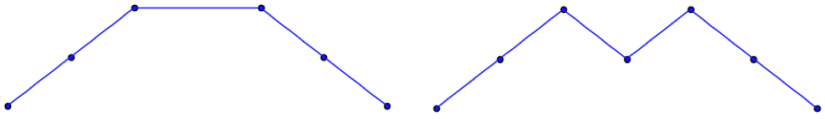
Definition 3. We define a relation *TwigR* on the Schröder paths. Suppose A and B are Schröder paths represented as strings.

- If we replace S in A with DU and that string equals B, then $A \to B$.
- Let C be a valid Schröder path represented as a string. If we replace UCD in A with either SC or CS and that string equals B, then $A \to B$.

1.3 Rule Examples

Consider strings $A = UUSDD$ and $B = UUDUDD$.

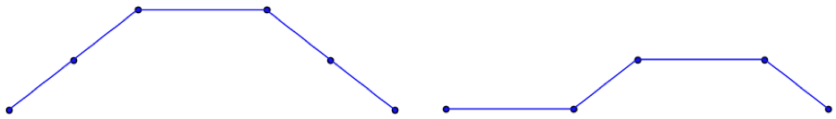
Figure 2: Path A (left) and Path B (right)



Therefore, $A \to B$ since the S in position 3 of A is replaced by DU in positions 3 and 4 of B.

Consider strings $A = UUSDD$, $B = SUSD$, and $C = USD$.

Figure 3: Path A (left) and Path B (right)



Therefore, $A \to B$ since UCD in A is replaced by SC in B.

Note: Multiple replacements cannot be completed at once.

Consider strings $A = USSD$, $B = UDUDUD$.

Therefore, there is not a relation $A \to B$ since multiple replacements must be completed (S in position 2 of A is replaced by DU in positions 2 and 3 of B and S in position 3 of A is replaced by DU in positions 4 and 5 of B.

2 Foundations of Counting Paths

Lemma 1. For any directed graph, all pairs of vertices, u and v, the u, v entry in X^r represents the number of paths from u to v with length r.

Note: We prove the Lemma proposed by Godsil and Royle (2013).

Proof. We will use proof by induction. Therefore, we assume

$$D \text{ is an arbitrary directed graph} \tag{1}$$

$$\text{with adjacency matrix } X, \tag{2}$$

and

$$u \text{ and } v \text{ are arbitrary vertices in } D. \tag{3}$$

We will show that X^r for all natural numbers r represents the number of paths from vertex u to vertex v with length r using induction.

Basis step: Consider $r = 1$. From statement (2), it follows that, since $X^1 = X$,

$$\text{X counts the length 1 paths,} \tag{4}$$

by definition of an adjacency matrix. Therefore, since $u, v \in D$ by statement (3), we find the u, v entry in X is the number of length 1 paths from u to v.

Induction step: We assume for some natural number k that

$$X^k \text{ counts length } k \text{ paths.} \tag{5}$$

We will show that X^{k+1} counts length $k + 1$ paths.

Consider $X^k + X$. By statements (4) and (5), we know that X^k and X count length k and 1 paths, respectively. Therefore, it follows that $X^k + X = X^{k+1}$ counts all the length $k + 1$ paths.

□

Lemma 2. For any acyclic directed graph, all pairs of vertices, u and v, the u, v entry in $(I - X)^{-1}$ represents the number of paths from u to v.

Proof. We make the same three assumptions in statements (1), (2), and (3). We make an additional assumption that D is acyclic. We will show that $(I - X)^{-1}$ represents the number of paths from vertex u to vertex v.

Consider $I - X$. By statement (1), D contains no cycles. In turn, we know X is upper triangular with all zeroes on the main diagonal. Therefore, $I - X$ has all zeroes in its lower triangle and ones on its main diagonal. It directly follows that $I - X$ is invertible.

By Lemma 1, we know for any natural number n, X^n counts length n paths. As such, we assume arbitrary matrix S, which counts all paths, such that

$$S = \sum_{m=1}^{\infty} X^m = I + X + X^2 + X^3 + ... \tag{6}$$

Consider S_m, the sum of the first m terms in the series for some natural number m. We can see that

$$S_m(I - X) = I - X^{m+1} \tag{7}$$

and

$$(I - X)S_m = I - X^{m+1} \tag{8}$$

Since for X^m there exist no paths of length k or greater, for natural number k representing the number of vertices in D, we can see that

$$\lim_{l \to \infty} X^m = 0. \tag{9}$$

In turn, we can observe from statements (7), (8), and (9) that

$$S(I - X) = I \tag{10}$$

and

$$(I - X)S = I \tag{11}$$

Therefore, using statements (10) and (11), we conclude that the the series converges to

$$\sum_{m=1}^{\infty} X^m = (I - X)^{-1} \tag{12}$$

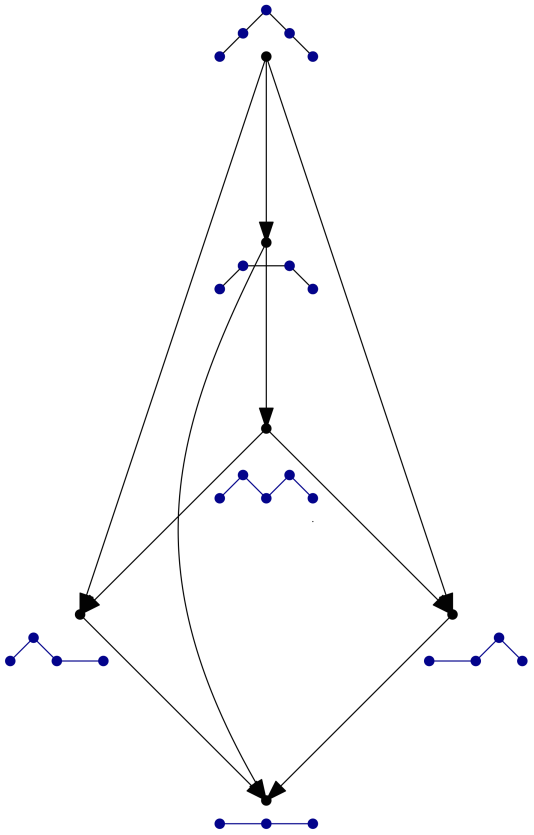
In turn, by equation (12), we have shown that $(I - X)^{-1}$ counts all paths in D. Again, since $u, v \in D$ by statement (3), we find the u, v entry in $(I - X)^{-1}$ is the number of any length path from u to v.

□

3 Path Counting Intuition

Using our intuition gained from Lemmas 1 and 2, we explore an example of Large Schröder - Size 2. We can see in Figure 4 that each vertex in the digraph is a Schröder path with either exactly 2 up steps, 1 up step and 1 straight step, or 2 straight steps. An edge between vertex u and v in the digraph means there is a $TwigR$ from u to v.

Figure 4: Digraph of Size 2 Schröder Paths



In observing the following equation, we find that $(I - X)^{-1}$ counts all possible paths of any length in the Digraph on Size 2 Schröder Paths.

$$\left(I - \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \right)^{-1} = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 5 \\ 0 & 1 & 1 & 1 & 1 & 3 \\ 0 & 0 & 1 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

We can explore a more interesting example for Size 3 Small Schröder paths. In the fourth row of the last column, we find a bolded 3. This 3 can be visualized as the three paths in Figure 5.

$$\begin{bmatrix} 0 & 1 & 1 & 2 & 2 & 5 & 2 & 2 & 7 & 7 & 14 \\ 0 & 0 & 1 & 1 & 1 & 3 & 1 & 1 & 4 & 4 & 8 \\ 0 & 0 & 0 & 1 & 1 & 2 & 1 & 1 & 3 & 3 & 6 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 2 & 1 & \mathbf{3} \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

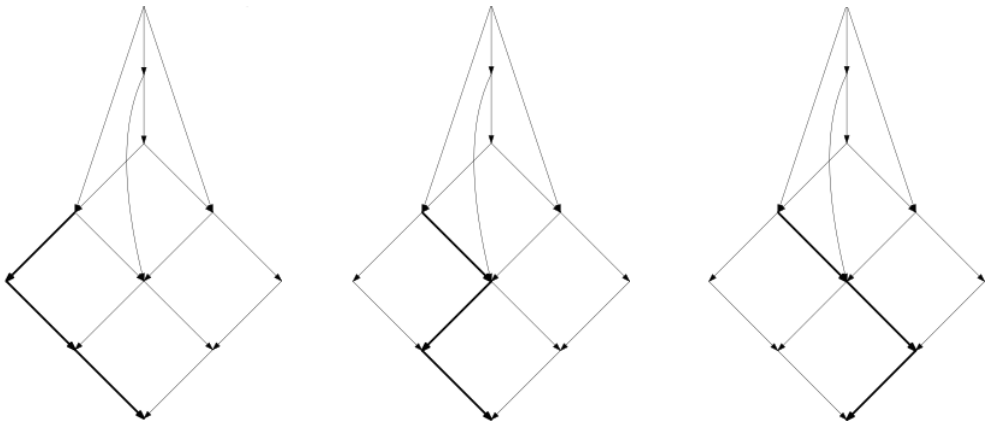
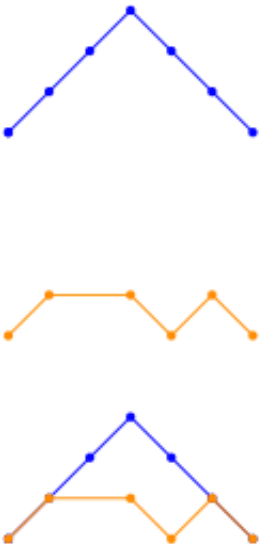


Figure 5: Three paths

4 Equivalency to Tilings

The rules for which $TwigR$ were defined may not be intuitive without context. Kenyon and Wilson (2011) developed a digraph on Dyck paths inspired by tilings. We developed $TwigR$ with two rules that formed natural tilings between upper and lower paths. Examples of these upper and lower paths can be seen in Figure 6.

(a) Upper and lower paths together form a space natural for tilings



(b) Digraph with paths corresponding to vertices

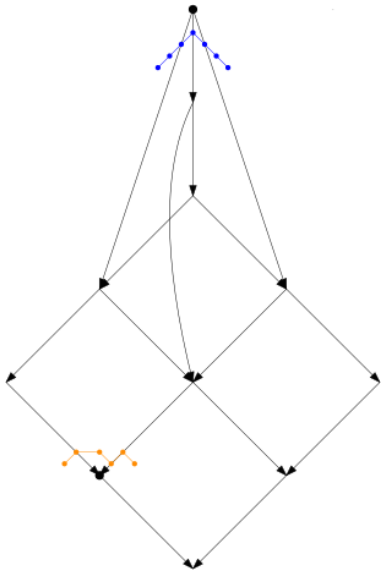
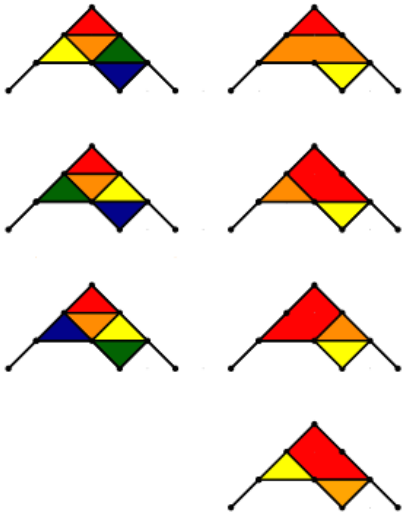


Figure 6: Example upper and lower paths

In the $(I - X)^{-1}$ matrix corresponding to the digraph on size 3 small Schröder paths, we find a 7 in the entry on the first row and 2nd to last column. As uncovered from Lemmas 1 and 2, this 7 represents the 7 possible paths from the corresponding vertices. We look to represent this 7 is another way through coloring of tilings as seen in Figure 7.

We note there are four length 3 paths in Figure 6b, which correspond to the four tilings of three colors. These tilings do not necessarily need coloring, but there is an inherent ordering in the coloring (red = first, orange = second, yellow = third, green = fourth, blue = fifth), which we want emphasize. However, unlike the Kenyon and Wilson (2011) result which found representations of the digraph on Dyck Paths using only tilings, we find three length 5 paths which correspond to the same tilings when coloring is ignored. In turn, we demonstrate a motivation for a representation of coloring and tilings on the digraph of Schröder paths.

Figure 7: Coloring of tilings on a pair of vertices for size 3 small Schröder



5 Exploration of Digraph Properties

In the creation of $(I - X)^{-1}$ matrices, we explored a handful of properties: maximal element and row/column sums. We first look at column and row sums for sizes 1 to 3 of large Schröder paths.

Size 1
Column Sums:

$$[1 \quad 2]$$

Row Sums:

$$[2 \quad 1]$$

Size 2
Column Sums:

$$[1 \quad 2 \quad 3 \quad 5 \quad 5 \quad 13]$$

Row Sums:

$$[12 \quad 7 \quad 5 \quad 2 \quad 2 \quad 1]$$

Size 3
Column Sums:

$$[1 \quad 2 \quad 3 \quad 5 \quad 6 \quad 8 \quad 5 \quad 13 \quad 20 \quad 31 \quad 6 \quad 20 \quad 41 \quad 82 \quad 54 \quad 170 \quad 8 \quad 31 \quad 82 \quad 191 \quad 170 \quad 607]$$

Row Sums:

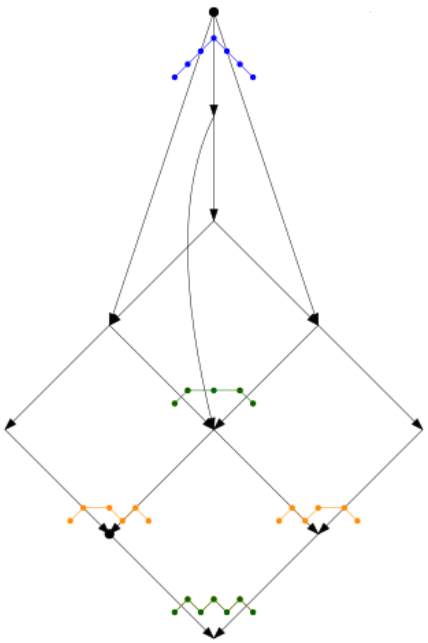
$$[537 \quad 296 \quad 227 \quad 108 \quad 49 \quad 12 \quad 108 \quad 54 \quad 26 \quad 7 \quad 49 \quad 26 \quad 16 \quad 5 \quad 5 \quad 2 \quad 12 \quad 7 \quad 5 \quad 2 \quad 2 \quad 1]$$

Based on the results from the column and row sums, we were unable to find any existing patterns when consulting The On-Line Encyclopedia of Integer Sequences (OEIS). However, we did observe an interesting pattern in repeated pairs of measurements in the row/column sums.

For example, there are repeated 5s in the column sums for size 2 and there are repeated 108s in the row sums for size 3. Through further exploration, we were able to deduce that these repetitions corresponded to symmetries in the digraph. In Figure 8, we can see that symmetries in the digraph correspond to path symmetries.

Can we count these? Future work includes exploring conjectures for asymmetric pairs of Schröder paths.

Figure 8: Symmetric paths (green) and Asymmetric Path Pairs (orange)



We observe the dimensionality of this problem in Table 1, as beyond size 3, these matrices cannot be calculated by hand. Because of this fact, we were motivated to explore larger sized digraphs and their properties through Python.

Size	Number of vertices	Maximal element
0	1	1
1	2	1
2	6	5
3	22	209
4	90	232,243
5	394	15,489,086,743
6	1806	103,884,239,105,712,928

Table 1:
Exploring for size (number of U/S steps) dimensionality and maximal element of $(I - X)^1$

6 Implementation

The core contribution of this work is twofold: developing a new mathematical object–digraph on Schröder paths and developing a code-base that allows for the efficient exploration of digraphs that grow at an exponential rate. Although the rules that build $TwigR$ are naturally recursive, we opted to implement an iterative solution. This solution allows us to generate every possible path permutation and filter the desired paths using our set of rules. In turn, if we wanted to explore additional rules, change rules, or different types of paths, we could easily adapt the code-base to do so.

To better understand the core code-base, I have included snippets as the following figures. The entire Python code-base can be found at the following Github repository https://github.com/adamtwig/MTH_496.

Figure 9: Generation Step

```
# only operations are up, down, straight
operations = ['U','D','S']

# max string length is twice the path size (all UD steps)
maxPathLen = 2 * exampleKey + 1

# initialize data structures
possiblePaths = []
pathDict = {}
lexPathDict = {}

# generate all possible strings
for m in range(maxPathLen):
    possiblePaths = possiblePaths + [''.join(i) for i in it.product(operations, repeat = m)]

# filter out only paths we are interested in
for path in possiblePaths:

    if isValidPath(path):
        size = getPathSize(path)

        # key to dict is size (collect all paths for specific size)
        if size not in pathDict:
            pathDict[size] = [path]
        else:
            pathDict[size].append(path)
```

We can see the generation step in Figure 9, which depends upon the isValidPath helper function, generates all permutations and filters out invalid paths (those that have unbalanced U/D steps) and those that go below the x-axis. The isValidPath helper function can be seen in Figure 10.

Figure 10: isValidPath helper function

```
def isValidPath(myPath):
    """
    @summary: returns 0/1 (True/False) if path is valid
        -- makes sure path is balanced with U and D steps
        -- makes sure path does not go below x axis
    """

    unbalanced = 0
    belowAxis = 0
    valid = 0
    for char in myPath:
        if char == 'U':
            unbalanced +=1
        if char == 'D':
            unbalanced -=1
        if unbalanced < belowAxis:
            belowAxis = unbalanced

    if (unbalanced == 0) and (belowAxis == 0):
        valid = 1

    return valid
```

Our next step is to apply the rules to all the valid paths using our helper functions and keep track of each edge that the rule is applied. Once we've accumulated all the edges, then we map each path to its lexicographical ordering to obtain a upper triangular matrix. Then we iterate over every edge and set the value of the matrix for corresponding indices of that edge to be 1. Our final step is taking the difference from the identity and inverting which are accomplished using efficient numpy routines. All of these steps can be seen in Figure 11.

Figure 11: Steps to reach $(I - X)^{-1}$

```
lexList = []
edgeList = []
# map paths to lexiographic notation
for path in pathDict[exampleKey]:
    pathEdgeList = ruleStoDU(path) + ruleUDtoS(path)
    for vertex in pathEdgeList:
        edgeList.append((lexMapPath(path),vertex))
    lexList.append(lexMapPath(path))

# sort paths based on lexiographic ordering
sLexList = sorted(lexList)

# initialize adj matrix to all 0s
lexMatrix = np.zeros([len(lexList),len(lexList)], dtype=np.int)

# get vertex indices and set path between them
for i in range(len(edgeList)):
    vertex0index = sLexList.index(edgeList[i][0])
    vertex1index = sLexList.index(edgeList[i][1])
    lexMatrix[vertex0index,vertex1index] = 1

# get diff between identity and adj matrix
diffIandX = np.identity(len(lexList)) - lexMatrix

# take inverse of matrix (we know it's invertibile
#   because diagonal is all 1s and lower triangle is all 0s
lexInvMatrix = np.linalg.inv(diffIandX).astype(np.int)
```

We have two helper functions for each of the rules in $TwigR$. These functions can be seen in Figures 12 and 13.

Figure 12: Rule 1

```
def ruleStoDU(myPath):
    """
    @summary: turns straight steps into down-up
    """
    StoDUlist = []
    for i in range(len(myPath)):
        if myPath[i] == 'S':
            newPath = myPath[:i] + 'DU' + myPath[i+1:]
            if isValidPath(newPath):
                StoDUlist.append(lexMapPath(newPath))

    return StoDUlist
```

Figure 13: Rule 2

```
def ruleUDtoS(myPath):
    """
    @summary: turns up-down pairs into straight steps
    """
    UDtoSlist = []
    Ulist = []
    for i in range(len(myPath)):
        newPath = myPath
        # create a list of where up steps are
        if myPath[i] == 'U':
            Ulist.append(i)
        # as you find down steps, pop off corresponding up steps
        if myPath[i] == 'D':
            currU = Ulist.pop()
            currD = i
            stringBeforeU = myPath[:currU]
            stringBetweenUandD = myPath[currU+1:currD]
            stringAfterD = myPath[currD+1:]

            # create new path by adding straight step to left of remaining path
            newPathBefore = stringBeforeU + 'S' + stringBetweenUandD + stringAfterD
            newPathAfter = stringBeforeU + stringBetweenUandD + 'S' + stringAfterD

            if isValidPath(newPathBefore) and newPathBefore == newPathAfter:
                UDtoSlist.append(lexMapPath(newPathBefore))
            else:
                if isValidPath(newPathBefore):
                    UDtoSlist.append(lexMapPath(newPathBefore))
                if isValidPath(newPathAfter):
                    UDtoSlist.append(lexMapPath(newPathAfter))

    return UDtoSlist
```

7 Future Work

- Further explore symmetries conjectures for Schröder paths
- Better characterize tilings with coloring
- Explore conjectures for Small Schröder paths and Motzkin paths
- Strengthen Python code to allow for additional portability

References

Godsil, C., and Royle, G. F. 2013. *Algebraic graph theory*, volume 207. Springer Science & Business Media.

Kenyon, R. W., and Wilson, D. B. 2011. Double-dimer pairings and skew Young diagrams. *Electron. J. Combin.* 18(1):Paper 130, 22.